SCOPE: SCALING COUNTERFACTUAL, OBSERVATION-PRESERVING ENVIRONMENT FAMILIES FOR TERMINAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning on executable environments has become the main route to terminal agents, and scaling such environments is its precondition. Whatever their source, most existing efforts produce the same unit of data, a task with one solution verified in one initial state. Yet a successful run shows only that one plan worked in one state, not that the agent explored the state first. When several legal states fit its observations and it acts on one unchecked, the reward credits the run as fully as one that checked first. We term this shortfall exploration debt, and scaling by successful runs accumulates it rather than collects it. We present SCOPE, which scales environments around what a successful run leaves unresolved rather than around the runs a source yields. SCOPE locates the expansion point at which a frozen reference rests on facts its history has not fixed, grows a counterfactual family of initial states that reproduce the history yet break the continuation, certifies one blind check and one program that complete every member, and delivers each member from its root and from a frontier after the prefix, all by real execution and independent of any learner. The resulting SCOPE-4K holds 4,000 executable terminal training environments spanning 11 domains, with at least 1,024 counterfactual families at its core and at least 80% long-horizon tasks. Direct RL on this collection, with the verifier's reward alone, then collects the debt without any reward shaping. Extensive experiments on Terminal-Bench 2.1 and four held-out suites demonstrate that SCOPE is ❶ *effective*, lifting Qwen3.5-9B from 22.2% to 39.4% on Terminal-Bench 2.1 and surpassing the strongest constructed baseline by up to 6.3 points, ❷ *scale-consistent*, holding its gains from Qwen3.5-4B to 27B on one fixed collection, and ❸ *cost-efficient*, since neither a reminder, a fourfold inference budget, nor a matched construction budget closes the gap. Our code is available here, and SCOPE-4K is provided as supplementary material.

1 INTRODUCTION

Reinforcement learning (RL) has become a central route to capable LLM agents (Zhang et al., 2026; Xi et al., 2026; Feng et al., 2025). A policy acts in an interactive environment, receives a verifiable reward, and improves from its own rollouts, with gains reported across web navigation, tool use, software engineering, and terminal operation (Qi et al., 2025; Qian et al., 2025; Wei et al., 2025; Yang et al., 2026). Among these settings, the terminal agent has emerged as one of the most consequential. It operates a real computer through shell commands in a live file system and runtime, and its success is checked by executable tests, which makes terminal work at once broadly useful and precisely verifiable (Yang et al., 2023; 2024; Wang et al., 2025). What such an agent can learn is shaped by the environments it trains in. Each environment provides a task instruction, an executable initial state with its files and dependencies, and a verifier that judges the final state, and keeping the three consistent makes environments expensive to author and scarce (Merrill et al., 2026a; Yao et al., 2026). Scaling executable environments is therefore a precondition for scaling terminal agents.

Scaling terminal environments has accordingly become a research direction in its own right. By the source of a task, existing efforts fall into three lines. ❶ **Repository mining** turns real issues, commits, tests, and recorded runs into executable tasks with automated setup. Some pipelines curate training gyms, while others harvest tens of thousands of instances across languages, synthesize

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

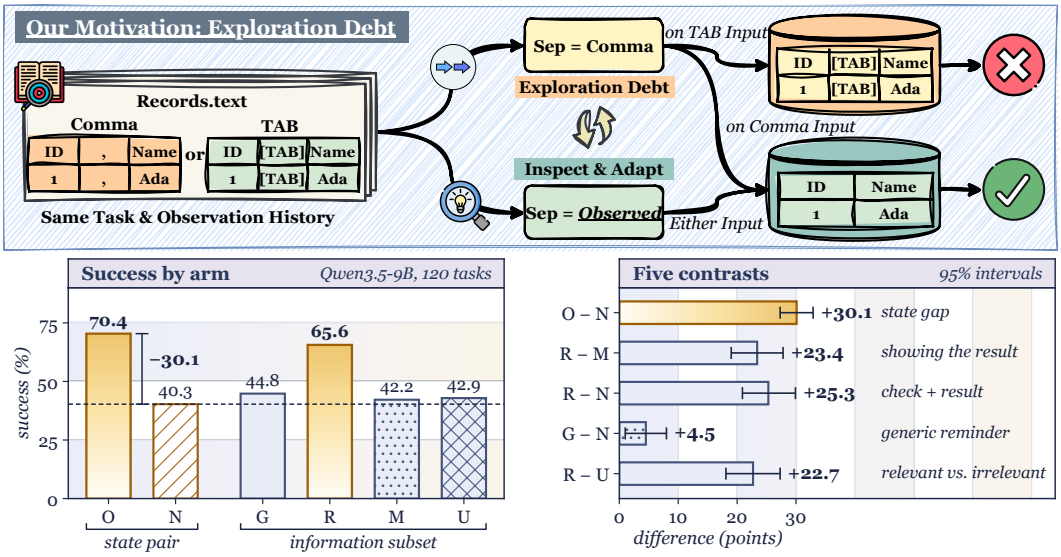


Figure 1: (***Top***) Motivation of SCOPE. (***Bottom***) Exploration debt of Qwen3.5-9B on 120 real tasks, with O the original state, N its rebuilt variant, G a generic reminder, R the relevant check revealed, M the check masked, and U an irrelevant check revealed (Appendix F).

further regressions inside the same repositories, or rebuild workspaces from agent trajectories (Pan et al., 2025; Yang et al., 2025; Badertdinov et al., 2026; Wu et al., 2026a). ❷ **Specification synthesis** builds tasks, environments, and verifiers together from domain specifications, capability taxonomies, or agent skills, and realizes them in containers (Hua et al., 2026; Cheng et al., 2026; Pan et al., 2026; Shen et al., 2026). ❸ **Task evolution** starts from a verified task and its reference solution and evolves it forward, extending it into longer workflows, rewriting it toward a target difficulty under verifier rewards, or calibrating it against solvers of known strength (Li et al., 2026; Wu et al., 2026c; Meng et al., 2026). Different as their sources are, these lines mainly check that each task admits a successful execution, and they scale by producing more such verified tasks.

However, a successful run shows only that its procedure worked in the state it was tested in, not that the agent explored the state first. Consider an agent asked to load a dataset into a database (Figure 1, Top). It imports the file with default settings and succeeds only because the file happens to be comma-separated. In an equally legal state where the file is tab-separated, the same command silently corrupts the table, and inspecting the file first would have revealed which case holds. Yet the reward credits this run as fully as an agent that checked first. We term this unresolved state dependence *exploration debt*, which arises when several legal states fit what the agent has observed and it acts on one of them without checking which holds. Exploration debt is acute in the terminal for three reasons. **I)** ***Pervasive.*** A terminal exposes little of its state unless asked, so a plan can rest on many unchecked facts and still succeed whenever they happen to hold. For 68.0% of the 241 real tasks that a post-trained model solves, a bounded search pairs its successful run with a legal state that reproduces every observation up to one point and defeats the plan after it (Appendix F). **II)** ***Consequential.*** An outcome reward credits a successful run as a whole without checking whether its procedure holds in other legal states, so the two behaviors look alike unless the data contains states that separate them. **III)** ***Substantial.*** As evidenced in Figure 1 (Bottom), success falls from 70.4% in the original state to 40.3% in the rebuilt one and recovers to 65.6% once the result of one ordinary check is placed before the model. These observations lead to our central research question:

RQ: *How can environment scaling collect exploration debt rather than accumulate it?*

To this end, we propose SCOPE, which scales environments around what a successful run leaves unresolved rather than around the runs a source yields. Starting from one success, SCOPE asks where the run acted on an unchecked fact and builds the states that would have exposed it, in three stages. **Expansion Point.** The run is frozen as a reference, and an LLM proposer and LLM judges locate the step at which its later decisions rest on facts the history has not fixed. **Counterfactual Family.** Around that step, an LLM edits the initial state, real execution keeps only the states that reproduce the history byte for byte yet defeat the continuation, and one blind check with one program,

confirmed to complete every member, certifies that the family is repaired by a single behavior rather than by luck. **Training from Roots and Frontiers.** Each member is delivered from its initial state and from a frontier right after the prefix, families are sampled in balance, and the learner is trained by direct RL on the verifier's reward alone. In this way, the outcome reward now tells checking from guessing, and the signal comes from the environments rather than from an added term.

Our contributions are summarized as follows:

- **Problem Identification.** We identify exploration debt, the state dependence a successful run leaves unresolved when several legal states fit its observations, and diagnose a recoverable adaptation gap in a post-trained model on a preregistered roster of 400 real terminal tasks.
- **Practical Solution.** We propose `SCOPE`, which expands a successful run at one selected point into a family of counterfactual environments that preserve its history, break its plan, and share one witnessed repair, so the outcome reward alone tells checking from guessing.
- **Empirical Validation.** Experiments on five suites show that `SCOPE` is ❶ ***effective***, lifting a 9B model from 22.2% to 39.4% on Terminal-Bench 2.1 and leading the strongest constructed baseline by 6.3 points, ❷ ***scale-consistent*** from 4B to 27B, and ❸ ***cost-efficient***, beating the baseline's 64k result at a 16k cap and leading at matched construction cost.
- **Open-Source Release.** We release `SCOPE-4K`, 4,000 executable terminal environments across 11 domains with 1,024 counterfactual families at its core, together with the construction pipeline, the evaluation rosters, and the complete analysis tables behind every result.

2 METHOD

As illustrated in Figure 2, `SCOPE` takes a successful run and returns a family of complete environments that share everything the run observed up to one expansion point, break its continuation, and are completed by one program after an ordinary check. For each successful run, it thus adds the states that expose what it left unchecked. We first formalize exploration debt (§2.1), then show how the expansion point is chosen (§2.2), how the family is grown and certified (§2.3), and how a learner trains from roots and frontiers (§2.4). Algorithm 1 (Appendix D) gives the full procedure.

2.1 PRELIMINARY

Notation. We consider a task $x = (C, \Omega, V_C)$ with a public specification C, an interaction protocol Ω that defines the available operations and their visible returns, and an outcome verifier V_C. A legal initial state $E \in \mathcal{E}(C)$ holds the files, inputs, dependencies, processes, and services the task involves. A program B is any procedure that acts through Ω, ranging from a fixed script to a policy that reads the state and branches on what it finds. Executing it from E yields a final state and a visible history, $\mathrm{Exec}_\Omega(E, B) = (S, h)$, where h records every action and its complete return. We write $\mathrm{State}_\Omega(\cdot, B)$ and $\mathrm{Hist}_\Omega(\cdot, B)$ for the two components, whether started from an initial or an intermediate state. The verifier $V_C(E, S) \in \{0, 1\}$ checks the final state against the public specification and accepts any solution that meets it. A learner π_θ is a program of this kind, and its trajectory τ receives the reward $R(\tau) = V_C(E, S_\tau)$, the verdict on its final state.

Exploration Debt. The learner is trained on a distribution $\mathcal{D}$ over environments to maximize

$$\max_\theta \ \mathbb{E}_{E\sim\mathcal{D}}\, \mathbb{E}_{\tau\sim\pi_\theta(\cdot|E)}\big[R(\tau)\big], \tag{1}$$

and what `SCOPE` designs is $\mathcal{D}$. Consider a run $\rho = P \cdot \sigma$ that succeeded on $E_0 \in \mathcal{D}$, split after a completed tool response into a prefix P and a continuation σ, with $\mathrm{Exec}_\Omega(E_0, P) = (S_0, h)$. Let A be an alternative continuation that first checks, from S_0, the fact that σ takes for granted and then adapts to it. On E_0 both continuations succeed, $V_C(E_0, \mathrm{State}_\Omega(S_0, \sigma)) = V_C(E_0, \mathrm{State}_\Omega(S_0, A)) = 1$, so they earn the same reward. Unless $\mathcal{D}$ contains a state that reproduces h and separates them, Eq. 1 cannot tell a policy that behaves like σ from one that behaves like A, an instance of the implicit partial observability that makes generalization in RL difficult (Ghosh et al., 2021). We say that the success of σ carries *exploration debt* under $\mathcal{D}$, a dependence on the state that the run leaves unresolved. More successful runs on E_0, or more tasks whose own solutions happen to work, do not by themselves supply a state that separates σ from A, which the rest of this section constructs.

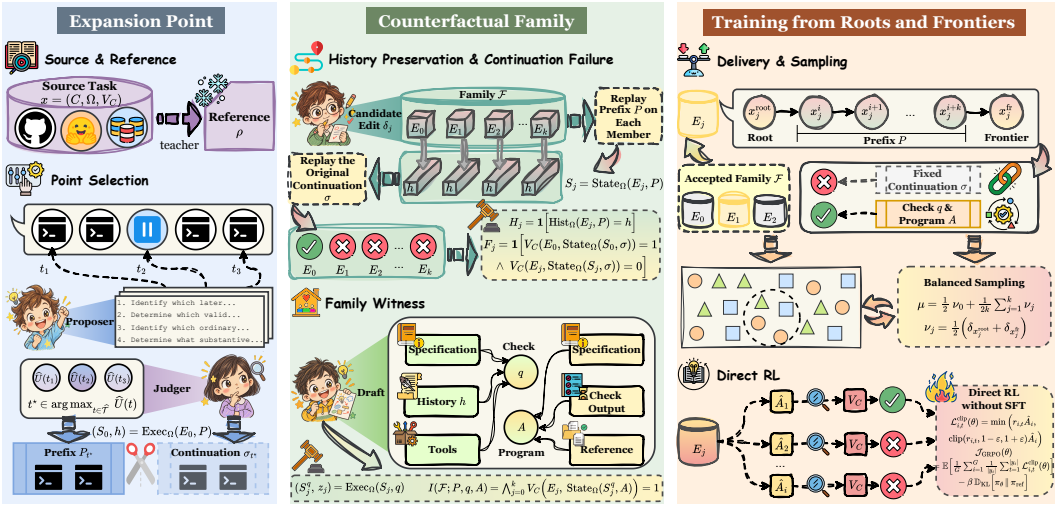


Figure 2: Overview of SCOPE.

2.2 Expansion Point

The expansion point is the step at which the reference splits into a prefix and a continuation, so it fixes which decision the family teaches. Whether a step is worth expanding depends on what the task leaves open there, a judgment a script cannot read off a log. We therefore select one point per source by LLM review and check the choice against the real execution of the reference.

Reference. Rather than a fixed dataset, SCOPE starts from sources. A source is any executable task with a verifier on which a successful run can be obtained, whether drawn from permitted public training tasks or built from real software projects (§3.1). For such a parent task, a teacher proposes an ordinary solution from the public task and the tools the environment provides, without access to any future edit or to the verifier's private answers. The first proposal accepted by independent verification is frozen as the reference ρ. Any step t that ends with a completed tool response splits it as $\rho = P_t \cdot \sigma_t$, the prefix holding substantive work and the continuation keeping its internal reads, branches, and recovery logic intact. Freezing the reference keeps the counterfactual aimed at a genuine successful procedure rather than at one weakened after the fact.

Point Selection. An LLM proposer reads the public task, the initial state, and the reference with its tool records, and proposes a few candidate steps. For each candidate, it answers four questions with evidence from the recorded run, namely **(i)** which later decisions rest on facts the history has not fixed, **(ii)** which legal edits to the initial state would leave that history intact, **(iii)** which ordinary observations would reveal those facts, and **(iv)** what substantive work remains after the step. LLM judges then check these answers and score each candidate's suitability. To limit the known biases of LLM judges (Zheng et al., 2023; Wang et al., 2024), each judge works in a fresh context over a shuffled candidate list, and a material disagreement triggers evidence-based adjudication or abstention (Appendix D). Scripts confirm only that each candidate step exists in the recorded execution. Let $\widehat{\mathcal{T}}$ denote the candidates that pass the judges and $\widehat{U}$ their suitability score. The expansion point is

$$t^\star \in \arg\max_{t \in \widehat{\mathcal{T}}} \widehat{U}(t), \tag{2}$$

and is frozen before any counterfactual is proposed. We write $P = P_{t^\star}$, $\sigma = \sigma_{t^\star}$, and $(S_0, h) = \mathrm{Exec}_\Omega(E_0, P)$. Each source is expanded at $t^\star$, and expansion at further points is studied as a control in Appendix G. In this way, the family diverges where the reference acts on an unchecked fact.

2.3 Counterfactual Family

For one parent task, SCOPE returns a counterfactual family of legal initial states

$$\mathcal{F} = \{E_0, E_1, \ldots, E_k\}, \qquad 1 \le k \le K_{\max}, \tag{3}$$

whose members are identical in everything the task makes public and differ only in internal facts that the specification leaves open, with $K_{\max}$ a cap rather than a number to fill and a single variant

the smallest case. A variant is worth adding only if **(i)** the prefix sees on it exactly what it saw on the original, so the history is preserved, **(ii)** the continuation fails on it, so the plan is broken, and **(iii)** one ordinary check and one program shared by the whole family complete it, so the break can be repaired by a behavior the learner could acquire. All three are settled by execution rather than by LLM judgment. We therefore grow the family by proposing edits around the frozen point and certify each condition by running it, H and F on every member and I on the family as a whole.

History Preservation and Continuation Failure. An LLM proposes candidate edits within a budget of J proposals per source. Each edit δ_j comes with a reason the specification permits it and a script that applies it while updating dependent objects consistently. The candidate $E_j = \delta_j(E_0)$ thus differs from the original only in private facts and is changed once, before any execution. Let $S_j = \mathrm{State}_\Omega(E_j, P)$ be the state the prefix reaches on it. The candidate is retained only if both

$$\begin{aligned} H_j &= \mathbf{1}\big[\,\mathrm{Hist}_\Omega(E_j, P) = h\,\big], \\ F_j &= \mathbf{1}\big[\,V_C\big(E_0, \mathrm{State}_\Omega(S_0, \sigma)\big) = 1 \;\wedge\; V_C\big(E_j, \mathrm{State}_\Omega(S_j, \sigma)\big) = 0\,\big] \end{aligned} \tag{4}$$

hold, where H_j, compared byte for byte, confines the edit to what the prefix did not look at and F_j asks only that this one continuation fail, not that the variant be unsolvable. A reference that already handles every examined variation yields no variant, and we do not weaken it to obtain one. Among candidates that pass, a new member is admitted only if it demands a different handling from those already retained, judged on the real state and the downstream effect.

Family Witness. Breaking the plan is not yet something to learn from. If every variant needed a fix of its own, the reward would punish σ on each of them without rewarding any single behavior in its place. The witness therefore certifies that one check and one program complete every member. We leverage an LLM to draft both, each from a restricted view. The check q sees only the public specification, the history h, and the tools, and nothing of the edits, the continuation, or the members, and it is not allowed to finish the task by itself. The program A sees the specification, the check outputs, and the reference. It is one program whose behavior verifiably depends on what the check reports rather than a selector between prepared answers keyed by a private label, and it may keep querying, verifying, and recovering as it runs. The check runs from each post-prefix state and the program from the checked state, and the witness holds when

$$\begin{aligned} (S_j^q,\, z_j) &= \mathrm{Exec}_\Omega(S_j, q), \qquad j = 0, \ldots, k, \\ I(\mathcal{F}; P, q, A) &= \textstyle\bigwedge_{j=0}^{k} V_C\big(E_j,\, \mathrm{State}_\Omega(S_j^q, A)\big) = 1, \end{aligned} \tag{5}$$

where z_j is the observation the check returns and S_j^q the state it leaves behind. Equal observations across members are acceptable as long as they call for the same processing. I is verified on two fresh rebuilds of every member, and a candidate enters the family only if the expanded group still passes. A family that passes H, F, and I thus pairs a plan that fails on every retained variant with one program that reads the check and completes every member, the original included.

2.4 ▶ Training from Roots and Frontiers

A family becomes training data only once a learner can reach its unresolved decision and the reward can act on it, which raises two difficulties. ❶ A learner that starts from the root follows its own trajectory and may end before the relevant decision, and ❷ a family sampled by its size lets prolific sources and repeated originals dominate the update. We therefore deliver every member from both its root and its frontier, sample families in balance, and train with standard GRPO.

Delivery and Sampling. Accepted families, those with at least one certified variant, constitute the core collection $\mathcal{K}$. Together with shared background tasks and control tasks that carry no counterexample, $\mathcal{K}$ forms the training distribution $\mathcal{D}$. The collection is built without access to any learner and shared unchanged across learners. To address ❶, each member E_j is instantiated as two training environments, a root $x_j^{\mathrm{root}} = E_j$ and a frontier $x_j^{\mathrm{fr}} = (S_j, h)$, where S_j is the state the common prefix P reaches on E_j and h the history it produces, identical across members by H_j. Frontiers are materialized before any of σ, q, or A is executed, so a learner started there faces the unresolved decision by construction, while the root keeps complete-task training. At a frontier, h is provided as context with zero weight in the policy loss, as are tool returns, while σ, q, and A remain hidden. The learner is free to inspect the state, act directly, or recover from an error. To address ❷, a parent

is drawn first, its original is balanced against the combined mass of its variants, and each member against its two starts,

$$\mu = \tfrac{1}{2}\,\nu_0 + \tfrac{1}{2k}\sum_{j=1}^{k}\nu_j, \qquad \nu_j = \tfrac{1}{2}\big(\delta_{x_j^{\mathrm{root}}} + \delta_{x_j^{\mathrm{fr}}}\big), \tag{6}$$

so neither a large k nor a repeated original inflates a source's weight. The rule constrains sampling, not storage, so environments may be shuffled once family, member, and start are recorded.

Direct RL. The reward remains the verifier's verdict $R(\tau) = V_C(E, S_\tau)$ of §2.1, with no shaping for the number of checks or for agreement with the reference. Since every accepted family passes F and I, the fixed continuation σ succeeds from the original frontier alone while the adaptive program A succeeds from every member, so under Eq. 6 their certified mean rewards from the frontiers are 0.5 and 1. As a result, Eq. 1, which could not separate the two on E_0, now favors A. We optimize the policy directly with GRPO (Shao et al., 2024; Guo et al., 2025), without a task-specific SFT stage. For each sampled member and start, a group of G rollouts is drawn and their rewards are normalized within the group into advantages $\hat{A}_i$, which are nonzero only when the rollouts of a group earn different rewards, giving the objective

$$\mathcal{J}(\theta) = \mathbb{E}\Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\big(r_{i,t}\hat{A}_i,\ \mathrm{clip}(r_{i,t}, 1-\varepsilon, 1+\varepsilon)\,\hat{A}_i\big)\Big] - \beta\,\mathbb{D}_{\mathrm{KL}}\big[\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big], \tag{7}$$

where y_i denotes the learner's tokens in rollout i, $r_{i,t}$ the token-level probability ratio, ε the clip range (Schulman et al., 2017), β the KL weight, and π_{ref} the initial policy. Exploration is thus rewarded by the environments themselves, not by an added term in the objective.

3 Experiments

In this section, we conduct extensive experiments to answer the following research questions ($\mathcal{RQ}$). ($\mathcal{RQ}$**1**) Is `SCOPE` effective on native long-horizon terminal tasks? ($\mathcal{RQ}$**2**) Do its gains hold across model scales? ($\mathcal{RQ}$**3**) Is `SCOPE` cost-efficient, or can cheaper alternatives replace it? ($\mathcal{RQ}$**4**) How sensitive is `SCOPE` to its key components and parameters, and what behavior does it instill?

3.1 Experimental Setup

Evaluation and Benchmarks. Our evaluation covers five suites from two groups, including ❶ *external* tasks on Terminal-Bench 2.1 (89 tasks) (Merrill et al., 2026a;b), run under its native harness and verifiers, and ❷ *held-out internal* suites built from sources disjoint from training, namely Unseen with 300 parents, Real with 150 independent workflows, Mechanism with 100 parents whose adaptation mechanisms are excluded from training, and Long with 120 parents at dependency depths 2, 4, and 6. Each task starts from the root, with no reference history, check, or program. We report the mean success rate (%) over three training seeds, each evaluated five times per task state, and its standard deviation across seeds. Differences between `SCOPE` and the strongest constructed baseline carry clustered bootstrap intervals (Agarwal et al., 2021).

Training Data. `SCOPE` and all constructed baselines start from the same sources, permitted public training tasks and real software projects whose workflows are lengthened along producer-consumer dependencies (Appendix D). On these sources, `SCOPE` yields `SCOPE-4K`, 4,000 executable environments across 11 domains, of which 3,400 form the core of 1,024 counterfactual families with up to 4 variants, 400 are shared background tasks and 200 shared controls, and 84% are long-horizon. An original is counted once, so the core size is $N_{\mathcal{K}} = \sum_i (1 + k_i)$, and extra starts or rollouts add no instances. An audit of 200 families finds that 98.0% reconstruct, the witness holds on 99.0% of those, and 1.0% of near-misses and 0.9% of legal alternatives are misjudged (Table 7).

Baselines. We compare `SCOPE` against five baselines, categorized into four groups. ❶ **Vanilla** is the public post-trained checkpoint without further training. ❷ **RL on original environments** continues RL on the parents alone and adds no state, namely Seed-RL. ❸ **State variation** adds legal variants that pass the quality checks in a fixed random order, namely Random-State. ❹ **Feedback-driven construction** lets the proposer see verification feedback when writing variants, including Feedback-State, which sees requirement-level feedback, and Trace-Feedback, which also sees the complete reference trace and is the strongest of them. For a fair comparison, the constructed baselines differ from `SCOPE` only in how variants are chosen, and all other settings are identical.

Table 1: Success rate (%) on five suites, as the mean over three seeds, its SD in gray (over five repeats for Vanilla), and its gain over Vanilla (unrounded). Best in **bold**, runner-up underlined.

Method	External	Held-out Internal			
	Terminal-Bench 2.1	Unseen	Real	Mechanism	Long
Qwen3.5-4B					
Vanilla	15.7 ±1.8	17.3 ±0.8	21.6 ±1.4	13.9 ±0.8	8.6 ±0.8
Seed-RL	20.1 ±0.9 ↑4.3	22.9 ±0.7 ↑5.6	27.5 ±1.5 ↑5.9	19.5 ±0.5 ↑5.6	15.0 ±0.4 ↑6.4
Random-State	22.3 ±0.9 ↑6.6	25.6 ±0.6 ↑8.3	29.2 ±1.4 ↑7.6	21.0 ±1.0 ↑7.1	17.4 ±0.7 ↑8.8
Feedback-State	23.5 ±1.3 ↑7.8	26.7 ±0.8 ↑9.4	30.0 ±1.5 ↑8.4	22.7 ±1.1 ↑8.8	19.1 ±0.8 ↑10.5
Trace-Feedback	24.3 ±0.8 ↑8.5	27.8 ±0.7 ↑10.5	31.4 ±1.2 ↑9.8	24.1 ±0.9 ↑10.2	20.4 ±0.6 ↑11.8
SCOPE (Ours)	**28.2** ±0.9 ↑12.5	**33.0** ±0.7 ↑15.7	**36.5** ±0.9 ↑14.9	**29.7** ±1.0 ↑15.8	**25.6** ±0.9 ↑17.0
Qwen3.5-9B					
Vanilla	22.2 ±2.2	23.1 ±0.9	27.2 ±1.7	18.4 ±1.2	12.7 ±0.9
Seed-RL	27.1 ±0.8 ↑4.9	29.1 ±0.6 ↑6.0	32.6 ±0.8 ↑5.4	24.0 ±0.9 ↑5.6	20.1 ±0.5 ↑7.4
Random-State	29.5 ±1.2 ↑7.3	32.5 ±0.7 ↑9.4	36.4 ±1.1 ↑9.2	28.0 ±1.0 ↑9.6	24.1 ±1.0 ↑11.4
Feedback-State	31.7 ±1.5 ↑9.4	34.6 ±0.9 ↑11.5	38.1 ±1.9 ↑10.9	30.7 ±1.5 ↑12.3	26.8 ±1.4 ↑14.1
Trace-Feedback	33.1 ±1.4 ↑10.9	36.4 ±0.7 ↑13.3	39.8 ±1.4 ↑12.6	33.1 ±1.5 ↑14.7	29.4 ±1.3 ↑16.7
SCOPE (Ours)	**39.4** ±1.3 ↑17.2	**42.1** ±0.8 ↑19.0	**45.3** ±1.2 ↑18.1	**39.2** ±1.0 ↑20.8	**35.1** ±1.0 ↑22.4
Qwen3.5-27B					
Vanilla	43.1 ±2.7	41.7 ±1.2	45.1 ±1.8	35.6 ±1.6	31.8 ±1.6
Seed-RL	45.4 ±1.4 ↑2.2	46.4 ±0.9 ↑4.7	49.0 ±1.0 ↑4.0	40.3 ±1.0 ↑4.7	36.5 ±0.7 ↑4.7
Random-State	47.0 ±2.1 ↑3.9	48.7 ±0.9 ↑7.0	51.4 ±1.5 ↑6.4	42.6 ±1.2 ↑7.0	39.1 ±1.0 ↑7.3
Feedback-State	47.9 ±2.2 ↑4.8	50.1 ±1.0 ↑8.4	53.2 ±2.4 ↑8.1	44.5 ±1.6 ↑8.9	40.6 ±1.3 ↑8.8
Trace-Feedback	49.1 ±2.1 ↑6.0	51.3 ±1.2 ↑9.6	54.8 ±1.3 ↑9.7	46.0 ±1.1 ↑10.4	42.2 ±1.6 ↑10.4
SCOPE (Ours)	**54.2** ±1.9 ↑11.1	**57.1** ±0.9 ↑15.4	**60.2** ±1.5 ↑15.2	**51.7** ±2.1 ↑16.1	**47.1** ±1.9 ↑15.3

Implementation Details. We instantiate SCOPE on Qwen3.5-4B, 9B, and 27B, three scales of one model family, each trained from its public post-trained checkpoint with standard GRPO and family-balanced sampling and no SFT stage. All five trained conditions run at every scale with three seeds each, giving 45 trained checkpoints. The main comparison uses 64M learner-generated tokens per run, and the analyses that retrain the model in §3.3 and §3.4 use 32M.

3.2 Main Results ($\mathcal{RQ}1$ & $\mathcal{RQ}2$)

Table 1 compares SCOPE with all baselines on five suites at three scales, and Figure 3 (Left) and Table 14 stratify the Long suite by depth. Obs. ❶–❸ answer $\mathcal{RQ}1$ and Obs. ❹ answers $\mathcal{RQ}2$.

Obs. ❶ SCOPE achieves the best result across all backbones and suites. As shown in Table 1, SCOPE outperforms all five baselines on every suite at every scale. On Terminal-Bench 2.1, it lifts Qwen3.5-4B, 9B, and 27B from 15.7/22.2/43.1% to 28.2/39.4/54.2% (12.5↑, 17.2↑, and 11.1↑). The margin over the strongest constructed baseline, Trace-Feedback, is 4.0/6.3/5.1 points, and the adjusted intervals of all nine primary contrasts exclude zero (Table 12). Notably, the gain is not confined to the state a plan happened to fit, as joint success on the Unseen sentinel pair rises and extends to every retained member (Appendix G). These results show that SCOPE is effective.

Obs. ❷ Collecting exploration debt helps more than adding runs of the same kind. Not all ways of adding environments pay off equally. RL on original environments gains the least, with Seed-RL raising Qwen3.5-9B on Terminal-Bench 2.1 by only 4.9 points over Vanilla (22.2% → 27.1%), since it adds more successful runs on the same states. State variation, which adds states with no relation to the run, gains 7.3 points, and feedback-driven construction is the strongest family, with Feedback-State and Trace-Feedback gaining 9.4 and 10.9 points. Even so, Trace-Feedback, which sees the complete reference, trails SCOPE on all five suites, including Mechanism (33.1% vs. 39.2%), whose adaptation mechanisms no method saw in training. In contrast, the variants of SCOPE are certified to preserve the history, break the plan, and share one repair, so the reward favors a plan that checks. SCOPE also leads six released training resources in every training route (Table 17). Overall, what a variant certifies matters more than how many variants are added.

Obs. ❸ The gains persist as real dependency depth grows. As shown in Figure 3 (Left), success falls with depth for all methods, as expected of work whose errors propagate. Specifically, with Qwen3.5-9B, SCOPE reaches 43.9/34.8/26.6% at depths 2, 4, and 6 against 39.1/29.1/20.0% for

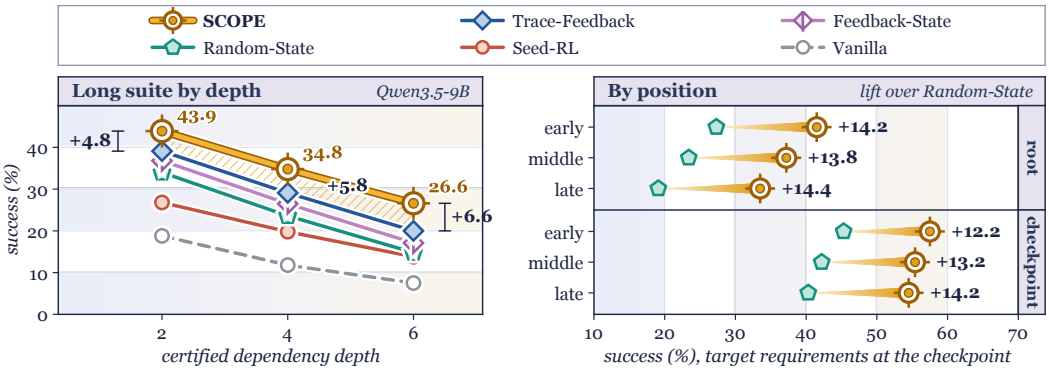


Figure 3: Success of Qwen3.5-9B by dependency depth (left) and by decision position (right).

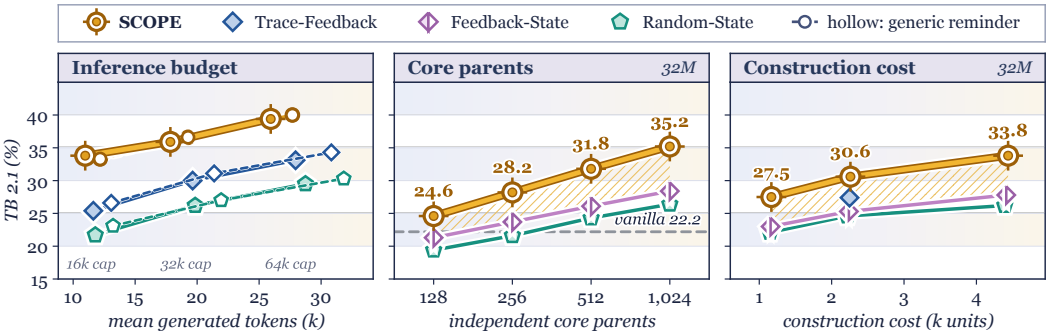


Figure 4: Cost analysis of SCOPE with Qwen3.5-9B on Terminal-Bench 2.1. Core parents are nested subsets of one collection, while each cost tier builds a separate collection, both at 32M tokens.

Trace-Feedback, and the margin grows from 4.8 to 5.8 and 6.6 points. The margin stays positive at every depth for Qwen3.5-4B and 27B as well, with a trend that varies across scales (Table 14). On the Long suite as a whole, SCOPE lifts Qwen3.5-9B from 12.7% to 35.1% (22.4↑), the largest gain in Table 1. This suggests that what is learned at one decision carries through the dependent work.

Obs. ❹ The gains of SCOPE hold across model scales and lift a compact model past its larger vanilla counterpart. On Terminal-Bench 2.1, the gain over Vanilla is 12.5, 17.2, and 11.1 points at the 4B, 9B, and 27B scales, and the margin over Trace-Feedback stays within 4.0∼6.3 points, all from one fixed collection. Notably, Qwen3.5-4B trained with SCOPE surpasses the vanilla Qwen3.5-9B on all five suites (28.2% vs. 22.2% on Terminal-Bench 2.1), and its gain (12.5 points) exceeds what moving the backbone from 4B to 9B brings (6.5 points). Within this model family, training the 4B model on SCOPE-4K thus brings more than moving it to 9B does.

3.3 Cost Analysis ($\mathcal{RQ}3$)

Figure 4 relates success to its inference and construction cost, and Obs. ❺–❻ answer $\mathcal{RQ}3$.

Obs. ❺ A reminder or a larger inference budget does not close the gap. We cross a default prompt with a generic inspection reminder and caps of 16k, 32k, and 64k generated tokens for the Random-State, Trace-Feedback, and SCOPE checkpoints. As shown in Figure 4 (Left), both help the baselines, as the reminder adds about 1 point and a fourfold cap 7.7∼7.8 points. Yet at the same 64k cap, Trace-Feedback with the reminder reaches 34.3%, below SCOPE without it (39.4%). Notably, SCOPE at 16k (33.8%) exceeds Trace-Feedback at 64k (33.1%) while generating 39.3% of its tokens. The reminder also helps SCOPE least, consistent with a behavior already learned (Table 18).

Obs. ❻ The advantage widens with more sources and holds at matched construction cost. As shown in Figure 4 (Middle), growing the core from 128 to 1,024 parents raises SCOPE on Terminal-Bench 2.1 from 24.6% to 35.2%, and its margin over Random-State and Feedback-State widens from 5.2/3.3 to 8.8/6.8 points. At matched construction cost with all failed work charged (Figure 4, Right), SCOPE leads these two by 4.5∼7.6 points at all three tiers and Trace-Feedback by 3.2 points at the middle tier. On fixed data, more RL compute keeps paying off (30.4% → 35.2% → 39.4% at 16M, 32M, and 64M). On the same sources, raising the variant cap from 1 to 4 adds 2.9 points, while more points per source lower success under one budget (Figure 9). Overall, certified counterfactuals cost more per accepted family than random variants yet pay off at matched cost.

Table 2: Ablation study of SCOPE with Qwen3.5-9B on Terminal-Bench 2.1.

Variant	Counterfactual Family				Expansion Point		Training			
	SCOPE	w/o H	w/o F	w/o I	SCOPE	random	SCOPE	flat, bal.	flat, unbal.	flat, par.
TB 2.1	**29.1**	23.6	22.2	21.3	**30.6**	26.7	**35.2**	32.8	30.6	35.0
Drop	–	↓5.5	↓6.9	↓7.8	–	↓3.9	–	↓2.4	↓4.6	↓0.2

3.4 Framework Analysis ($\mathcal{RQ}4$)

We analyze SCOPE along three lines, component ablation (Table 2), the use of information (Figure 5 and Table 25), and behavior (Figure 3, Right, and Tables 27, 28, and 29), which together answer $\mathcal{RQ}4$, and defer the sensitivity to its two caps to Appendix G (Figure 9).

Ablation Study. We ablate the three stages of SCOPE. For the counterfactual family, ❶ **w/o** H no longer requires the history to be preserved, ❷ **w/o** F no longer requires the continuation to fail, and ❸ **w/o** I drops the family witness, on a shared candidate pool with one variant per parent and matched members. For the expansion point, ❹ **random** takes one random legal step per source under the same budget. For training, ❺ **flat** sampling shuffles the same environments and ignores family membership, balancing the original against its variants overall, not at all, or per parent. As shown in Table 2, removing the witness causes the largest drop (29.1% → 21.3%), as the failure then has no verified shared repair to learn. Removing H or F costs 5.5 and 6.9 points, since without H the difference can show in the prefix and without F the continuation may still succeed. A random point costs 3.9 points. Flat sampling costs 2.4 and 4.6 points with overall and no balance, yet only 0.2 with equal parent weights, so parent-level weighting, not adjacency, is what matters.

Information Use. If SCOPE helps because the learner obtains and uses the relevant fact, handing that fact to a baseline should close most of the gap, and handing it an irrelevant one should not. On the Mechanism suite, we continue each learner from a common reference prefix under *natural* continuation, or run the relevant and an irrelevant check and show the relevant result (*reveal*), neither (*mask*), or only the *irrelevant* one. As shown in Figure 5, SCOPE leads Trace-Feedback by 9.8 points with Qwen3.5-9B and 11.5 with Qwen3.5-4B under natural continuation, yet by only 1.9 and 2.1 points under reveal, while mask and irrelevant stay near the natural level. Notably, on 400 tasks that require no discovery, SCOPE differs from Trace-Feedback by 0.3 points at a token ratio of 100.6%, within the prespecified margins. This suggests that the learner inspects where it pays rather than by habit.

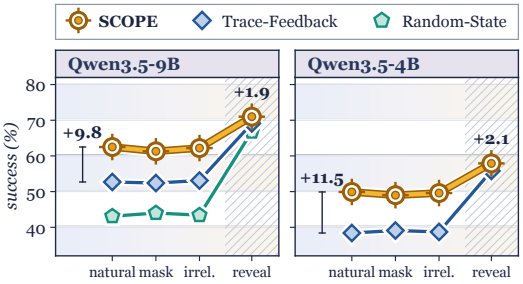


Figure 5: Interventions on the Mechanism suite.

Behavioral Internalization. We further examine what the learner does differently. First, the gain is not an opening routine, as on 60 matched blocks of seven-stage tasks whose unresolved decision arises after stage 1, 3, or 5, SCOPE leads Random-State by 13.8∼14.4 points from the root and by 12.2∼14.2 points on the target requirements from a common checkpoint (Figure 3, Right). Second, the learner commits fewer errors, as its first erroneous commitment rate on monitored runs falls from 32.2% to 19.9%, and from a shared recoverable error state, success without propagation rises from 35.0% to 55.2% (Table 27). Third, in an auxiliary SFT study, demonstrations that inspect before adapting beat direct continuations at all three scales (30.8% vs. 28.8% with 9B, Table 29).

4 Conclusion

In this paper, we identify exploration debt, the state dependence a successful run leaves unresolved when several legal states fit its observations, and diagnose it on a preregistered roster of real terminal tasks. To collect it, we propose SCOPE, which expands each successful run into a family of counterfactual environments that preserve its history, break its plan, and share one witnessed repair, so that the verifier's reward alone tells checking from guessing. Experiments on Terminal-Bench 2.1 show gains across model scales that hold under larger inference budgets and matched construction cost. We believe SCOPE paves the way for environment scaling that certifies what success leaves open.

AI USE STATEMENT

Generative AI tools are part of the method studied in this paper. As described in §2 and Appendix D, LLMs act as the teacher that proposes reference solutions, the planner and reviewers of expansion points, the author of candidate state edits, blind checks, and adaptation programs, and the offline judge that labels task type and difficulty, and GLM-5.3 serves as the demonstration teacher of the SFT studies in Appendix E.4 and Appendix E.5. Their outputs enter `SCOPE-4K` only after real execution and independent verification, and each role sees only the inputs listed in Table 6. In addition, we used generative AI tools to polish the writing of this manuscript. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. It involves no human subjects, crowdsourcing, or personal data. The environments are built from permitted public training tasks and open-source software projects, each registered with its version, ancestry, and usage terms, and the released environments follow the terms of the sources they derive from. All agent, reference, and verifier executions run in isolated environments, and the tasks concern ordinary software, data, and system maintenance. Terminal agents act on real computers, so stronger agents could also automate harmful or destructive operations if deployed without safeguards. We release `SCOPE-4K` and the pipeline for research use and recommend sandboxed execution and human oversight when agents trained on it act on real systems. To keep the evaluation honest, Terminal-Bench 2.0 and 2.1 tasks and their near-duplicates are excluded from all training sources (Appendix D).

REPRODUCIBILITY STATEMENT

We have made every effort to ensure the reproducibility of this work. §2 and Algorithm 1 specify the construction procedure, and Appendix D details the roles and their information boundaries, the proposal allowance and variant cap, the tests of H, F, and I, verifier qualification, frontier materialization, and the quality audit. Appendix C describes the composition of `SCOPE-4K`. Appendix E lists the evaluation suites, the baselines, the RL configuration with its seeds and budgets, the compute resources, the auxiliary SFT setup, the protocol for released training resources, and the statistical protocol, and Appendix F gives the protocol of the motivation study. Appendix G reports the full results with their intervals, and all results use public model checkpoints and a pinned revision of the external benchmark. The code of the construction pipeline, including resumable generation, is available at the anonymous repository linked in the abstract, and `SCOPE-4K` is provided as supplementary material.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Ibragim Badertdinov, Maksim Nekrashevich, Anton Shevtsov, and Alexander Golubev. SWE-rebench v2: Language-agnostic SWE task collection at scale. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=UCAda9kS57.

Zihao Cheng, Hongru Wang, Zeming Liu, Xinyi Wang, Xiangrong Zhu, Yuhang Guo, Wei Lin, Jeff Z. Pan, and Yunhong Wang. Terminal-world: Scaling terminal-agent environments via agent skills, 2026. URL https://arxiv.org/abs/2605.20876.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference*

on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 2020. URL http://proceedings.mlr.press/v119/cobbe20a.html.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, Shibin Wu, Zhengwei Tao, Yong Jiang, Pengjun Xie, Ningyu Zhang, Fei Huang, Wentao Zhang, and Jingren Zhou. Towards general agentic intelligence via environment scaling. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 17610–17621. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.872. URL https://doi.org/10.18653/v1/2026.findings-acl.872.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Kanishk Gandhi, Shivam Garg, Noah D. Goodman, and Dimitris Papailiopoulos. Endless terminals: Scaling rl environments for terminal agents, 2026. URL https://arxiv.org/abs/2601.16443.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why generalization in RL is difficult: Epistemic pomdps and implicit partial observability. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 25502–25515, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d5ff135377d39f1de7372c95c74dd962-Abstract.html.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Hao Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, Tao Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao

Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nat.*, 645(8081):633–638, 2025. doi: 10.1038/S41586-025-09422-Z. URL https://doi.org/10.1038/s41586-025-09422-z.

Zhanbo Hua, Yifan Yao, Weihao Xie, Yongchi Zhao, Minghao Liu, Ruizhi Qiu, Zhewei Huang, Zun Wang, Yiyan Ji, Yunhai Ye, Letian Zhu, Xinping Lei, Han Li, Zhiyuan Ma, Zili Wang, Zhaoxiang Zhang, and Jiaheng Liu. Cli-universe: Towards verifiable task synthesis engine for terminal agents, 2026. URL https://arxiv.org/abs/2606.22883.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents, 2026. URL https://arxiv.org/abs/2606.23321.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Zaid Khan, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. Dataenvgym: Data generation agents in teacher environments with student feedback. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=00SnKBGTsz.

Zhongzhi Li, Yucheng Shi, Zongxia Li, Ruhan Wang, Anhao Li, Zixun Huang, Junyao Yang, Lei Ke, Ninghao Liu, Haitao Mi, and Leowei Liang. Recursive synthesis for long-horizon terminal tasks, 2026. URL https://arxiv.org/abs/2608.05466.

Yusong Lin, Haiyang Wang, Shuzhe Wu, Lue Fan, Feiyang Pan, Sanyuan Zhao, and Dandan Tu. Cli-gym: Scalable cli task generation via agentic environment inversion, 2026. URL https://arxiv.org/abs/2602.10999.

Siyuan Lu, Zechuan Wang, Hongxuan Zhang, Qintong Wu, Leilei Gan, Chenyi Zhuang, Jinjie Gu, and Tao Lin. Don't just fine-tune the agent, tune the environment. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=nzodtGccEM.

Fanzhe Meng, Guoxin Chen, Jiale Zhao, Shuang Sun, Zhiyu Lin, Wayne Xin Zhao, Ruihua Song, Ji-Rong Wen, and Kai Jia. Calibforge: Adversarial solver calibration for scaling learnable terminal tasks, 2026. URL https://arxiv.org/abs/2608.06352.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong (Ryan) Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Christopher Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic

tasks in command line interfaces. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/444a3737adaee10d86ad2ef5f74468e6-Paper-Conference.pdf.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026b. URL https://arxiv.org/abs/2601.11868.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Zhihong Pan, Jiyuan He, Kai Zhang, Yupeng Han, Ze Liu, Yuze Zhao, Yongcong Ye, and Zhaohua Yang. Meta-task: Turning terminal task synthesis into a terminal task for scalable agent training, 2026. URL https://arxiv.org/abs/2607.27929.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob N. Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17473–17498. PMLR, 2022. URL https://proceedings.mlr.press/v162/parker-holder22a.html.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/294e09f267683c7ddc6cc5134a7e68a8-Abstract.html.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. Webrl: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego,*

CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/97c5b2707228e7e3fb67e4ecc2e0e607-Abstract-Conference.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Qijia Shen, Zhiqi Huang, Vamsidhar Kamanuru, Aznaur Aliev, Jay Rainton, Ahmed Awelkair, Zhichen Zeng, Jiajun Li, Shi Dong, Yueming Yuan, Boyuan Ma, Qizheng Zhang, Jiwei Fu, Yuzhen Mao, Wendong Fan, Ping Nie, Philip Torr, Bernard Ghanem, Changran Hu, Jonathan Lingjie Li, Urmish Thakker, and Guohao Li. Seta: Scaling environments for terminal agents, 2026. URL https://arxiv.org/abs/2607.10891.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö. Arik. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=3UKOzGWCVY.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. Os-genesis: Automating GUI agent trajectory construction via reverse task synthesis. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 5555–5579. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.277. URL https://doi.org/10.18653/v1/2025.acl-long.277.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 9440–9450. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.511. URL https://doi.org/10.18653/v1/2024.acl-long.511.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, and et al. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. SWE-RL: advancing LLM

reasoning via reinforcement learning on open software evolution. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/7107d4d2e837bde2171c6b71b5bde954-Abstract-Conference.html.

Jie Wu, Zhenru Zhang, Beichen Zhang, Xuwu Wang, Yuhui Su, Mouxiang Chen, Peng Wang, Zhihai Wang, Que Shen, Hao Zhou, An Yang, Fei Huang, Yujiu Yang, and Dayiheng Liu. Terminal-universe: Turning agent trajectories into scalable terminal environments, 2026a. URL https://arxiv.org/abs/2609.04148.

Siwei Wu, Yizhi LI, Yuyang Song, Wei Zhang, Yang Wang, Riza Batista-Navarro, Xian Yang, Mingjie Tang, Bryan Dai, Jian Yang, and Chenghua Lin. Large-scale terminal agentic trajectory generation from dockerized environments. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=PeFSCRulgy.

Xiaojun Wu, Cehao Yang, Honghao Liu, Xueyuan Lin, Zhichao Shi, Hao Zhou, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Envs-forge: Frontier-optimized reward-grounded environment synthesis for agent rl, 2026c. URL https://arxiv.org/abs/2608.14312.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, yajie yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. Agentgym-rl: An open-source framework to train llm agents for long-horizon decision making via multi-turn rl. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 12685–12727, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1571ce1ff735be4551db50887043726e-Paper-Conference.pdf.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=EEgYUccwsV.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/4b175d846fb008d540d233c188379ff9-Abstract-Datasets_and_Benchmarks.html.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated

software engineering. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/8b86cf5ace600c48fd188efbb8dedec8-Abstract-Datasets_and_Benchmarks_Track.html.

Junyao Yang, Yucheng Shi, Zhongzhi Li, Ruhan Wang, Zongxia Li, Haitao Mi, and Leowei Liang. T1: Terminal agent reinforcement learning for long-horizon tasks, 2026. URL https://arxiv.org/abs/2609.11042.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yihang Yao, Bo Pang, Xuan Phi Nguyen, Ding Zhao, Shafiq Joty, and Semih Yavuz. Learning generalizable behaviors for terminal agents, 2026. URL https://arxiv.org/abs/2608.22631.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhong-Zhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for llms: A survey. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=RY19y2RI1O.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Kaijie Zhu, Yuzhou Nie, Yijiang Li, Yiming Huang, Jialian Wu, Jiang Liu, Ximeng Sun, Zhenfei Yin, Lun Wang, Zicheng Liu, Emad Barsoum, William Yang Wang, and Wenbo Guo. Termigen: High-fidelity environment and robust trajectory synthesis for terminal agents, 2026. URL https://arxiv.org/abs/2602.07274.

A Related Work

Terminal Agents. Terminal agents act on a real computer through shell commands and are judged by what their actions leave behind. By contribution, prior work falls into three families. ❶ **Executable benchmarks** grade an agent by running its result, from repository issues and command-line workflows to web and desktop environments with persistent state (Jimenez et al., 2024; Merrill et al., 2026a; Zhou et al., 2024; Xie et al., 2024). ❷ **Scaffolds and inference-time procedures** improve how a fixed model interacts, through interleaved reasoning, software-oriented interfaces, open platforms, and verbal reflection (Yao et al., 2023; Yang et al., 2024; Wang et al., 2025; Shinn et al., 2023). ❸ **RL training** optimizes the policy itself on executable environments, from software evolution and multi-turn frameworks with fine-grained credit (Wei et al., 2025; Xi et al., 2026; Feng et al., 2025) to full recipes for long-horizon terminal RL (Yang et al., 2026; Yao et al., 2026). `SCOPE` keeps the scaffold, the optimizer, and the evaluation fixed and changes only the environments a learner trains on.

Environment Scaling. Executable environments are the scarce input of agentic RL, and scaling them has become a research direction of its own. By what is scaled, prior work falls into four lines. ❶ **Repository mining** turns real issues, commits, tests, and recorded runs into executable tasks with automated setup (Pan et al., 2025; Yang et al., 2025; Badertdinov et al., 2026; Wu et al., 2026a). ❷ **Specification synthesis** builds tasks, environments, and verifiers together from domain specifications, taxonomies, or agent skills (Hua et al., 2026; Cheng et al., 2026; Pan et al., 2026; Shen et al., 2026), and its counterparts for web, GUI, and tool agents synthesize tasks and trajectories from tutorials, reverse task synthesis, interaction, or simulated tools (Xu et al., 2025; Sun et al., 2025; Su et al., 2025; Fang et al., 2026). ❸ **Task evolution and learner-adaptive design** extend or rewrite verified tasks toward a target difficulty (Li et al., 2026; Wu et al., 2026c; Meng et al., 2026) or condition the next instance on the student through teacher agents, environment-side tuning, and regret-based curricula (Khan et al., 2025; Lu et al., 2026; Dennis et al., 2020; Parker-Holder et al., 2022), so that the output is a curriculum tied to one learner. ❹ **State variation** varies states to study transfer, augments transitions counterfactually, and explains through implicit partial observability why unresolved uncertainty makes information seeking matter (Cobbe et al., 2020; Pitis et al., 2020; Ghosh et al., 2021). These lines vary tasks, states, and curricula in different ways. We instead identify exploration debt and scale environments around it, certifying legal states that preserve a reference prefix, defeat its continuation, and admit one shared adaptive procedure.

B Notations

Table 3 summarizes the notation used in §2.

C Dataset Details

In this section, we describe `SCOPE-4K`, the collection that `SCOPE` builds from the sources of §3.1, including its composition, labels, domains, per-instance assets, and release format.

Composition. `SCOPE-4K` holds 4,000 unique executable environments in three mutually exclusive parts (Table 4). ❶ The **core** $\mathcal{K}$ holds 3,400 environments in 1,024 certified counterfactual families, of which 163, 527, 177, and 157 retain $k = 1$, 2, 3, and 4 variants, so that $N_{\mathcal{K}} = \sum_i (1 + k_i) = 3{,}400$. ❷ The **background** part $\mathcal{B}$ holds 400 environments in 160 sampling groups, 80 of two environments and 80 of three, that broaden the training distribution and carry no counterexample. ❸ The **control** part $\mathcal{U}$ holds 200 auxiliary environments in 80 groups, 40 of each size. The collection therefore has 1,264 sampling groups, of which 1,024 are certified families. The root and frontier starts of a member share one environment identity and add no environment, and reference or alternative executions add none either. The 85/10/5 split is an instance ratio, whereas the 70/20/10 ratio of Appendix E.3 is a sampling weight, and the two are not interchangeable.

Labels. Horizon and environment complexity are separate annotation axes. 3,364 environments carry the long-horizon label, with 174 medium and 462 short, and 3,424 carry the complex-environment label, with 576 simple. Their intersection, the long complex environments, holds 3,324

Table 3: Notations used in this paper.

Symbol	Meaning
$x = (C, \Omega, V_C)$	a task with its public specification, interaction protocol, and outcome verifier
$E \in \mathcal{E}(C)$, E_0, E_j	a legal initial state, the original environment, and the j-th variant
$\mathrm{Exec}_\Omega(E, B) = (S, h)$	final state and visible history of running a program B from E
$\rho = P \cdot \sigma$	the frozen reference run, its prefix, and its continuation
$t^\star$, $\widehat{\mathcal{T}}$, $\widehat{U}$	the expansion point, the candidate steps that pass review, and their suitability score
(S_0, h), (S_j, h)	the frontier state and history that P reaches on E_0 and on E_j
δ_j, J	a candidate edit of the initial state and the proposal allowance per source
$\mathcal{F} = \{E_0, \ldots, E_k\}$, $K_{\max}$	a counterfactual family with k variants and the variant cap
H_j, F_j	history preservation and continuation failure of member j (Eq. 4)
q, A, (S_j^q, z_j)	the blind check, the common program, and the state and observation the check leaves on member j
$I(\mathcal{F}; P, q, A)$	the family witness (Eq. 5)
$\mathcal{K}$, $\mathcal{D}$	the core collection of accepted families and the training distribution
x_j^{root}, x_j^{fr}	the root and frontier starts of member j
μ, ν_j	the family-balanced sampling distribution and the two-start mixture of member j (Eq. 6)
π_θ, π_{ref}, $R(\tau)$	the learner, its frozen initialization, and the verifier reward of trajectory τ
G, $\hat{A}_i$, $r_{i,t}$, ε, β	GRPO group size, advantage, ratio, clip range, and KL weight (Eq. 7)

Table 4: Composition of `SCOPE-4K`. Shares are over the 4,000 released environments, and weights are those of the training mixture.

Part	Environments	Groups	Share	Weight	Content
Core $\mathcal{K}$	3,400	1,024	85%	70%	certified counterfactual families, one original and $1 \le k_i \le 4$ variants each
Background $\mathcal{B}$	400	160	10%	20%	shared background groups of two or three environments
Control $\mathcal{U}$	200	80	5%	10%	shared auxiliary control groups of two or three environments
Total	4,000	1,264	100%	100%	unique executable environments, each with a task, an initial state, construction evidence, and a verifier

environments, and the remaining 676 are focused tasks. Difficulty is labeled easy for 418, moderate for 1,816, hard for 1,482, and very hard for 284 environments. The labels come from an offline judge that reads the public instruction, the dependency structure, the reference workload, and the recovery requirements, and a reviewer checks the labeling evidence (Appendix D).

Domains. The collection covers 11 domains (Table 5). Every domain holds at least 80 distinct families, of which at least 20 are long complex core families, and different scenarios, sources, and dependency structures within a domain count rather than renamed variables. Domain labels may overlap, and cross-domain workflows overlap by definition, so the per-domain counts sum to more than the 1,264 unique groups. No single project contributes more than 28 families, and the largest cluster of near-duplicate candidates holds 4 families. The eleven domains are a coverage vocabulary rather than a closed taxonomy of terminal skills.

Workflow Shapes. We distinguish three shapes. ❶ **Focused tasks** keep basic skills and difficulty diversity with real verifiable work. ❷ **Long complex tasks within one type** complete several dependent stages inside one area of work, such as a multi-stage software repair. ❸ **Long complex tasks across types** join different sub-tasks through real data or control dependencies, such as repairing a parser and then restoring the data it feeds, or migrating a database and verifying the service on top

Table 5: Domain coverage of SCOPE-4K. Families are counted per domain label, so the totals exceed the 1,264 unique groups.

Domain	Families	Long core	Domain	Families	Long core
Software maintenance	184	132	Scientific computing	129	89
Build and release	137	92	Machine learning	121	84
Files and storage	158	106	Media	108	72
Data engineering	236	174	Virtualization	86	57
Databases	161	118	Cross-domain workflows	128	104
Services and networking	172	124			

of it. The collection holds 3,324 long complex environments, 384 of them cross-type compositions, and 676 focused ones. Shape is a separate axis from the three parts above, so the core is not by definition long and the background is not by definition short, and shape is not a difficulty label.

Per-Instance Assets. Every environment ships four parts. ❶ **Task**, a public natural-language goal with the permitted operations and the success condition, identical word for word across the members of a family. ❷ **Environment**, a rebuildable complete initial state with the files, dependencies, and services the task needs, bound to its root and frontier starts. ❸ **Construction evidence**, the reference, the legal alternatives, the common check and program, and the execution records that certify the family. ❹ **Verifier**, an executable check of the final state against the public goal that accepts legal alternatives and does not grade report text, file names, or keywords. A family further carries stable family, member, and point identifiers, the original-to-variant relation, the shared history, and the root-to-frontier mapping. Construction evidence is an audit asset and stays out of the learner's view.

Release. The release separates execution assets from research-side evidence. Execution assets are the public instructions, the complete environments of all members, both starts, the public histories, and the collection and verification protocols. Research-side evidence covers the reference executions, the expansion-point candidates and reviews, all state candidates, the gate outcomes, the witness versions, the quality challenges, and the source and coverage records. Directory indices, image archives, and content checksums keep references reachable and versions consistent, and a checksum is not treated as evidence of semantic correctness. Generation is resumable, reuses assets that are complete and consistent, and repairs targeted failures without resampling accepted members. Cost accounting covers point planning and review, proposals, formatting retries, program repairs, reference and verification executions, failed candidates, and the preparation of both starts (Appendix G.2).

D Construction Details

In this section, we detail the construction pipeline of §2. Algorithm 1 gives the full procedure, Table 6 lists the roles, and the paragraphs below specify each stage.

Roles and Information Boundaries. Every generation and review role sees a fixed set of inputs and nothing else, so that no role can leak a private answer into the collection (Table 6). Prompts are fixed per role and describe the task, the visible inputs, the output constraints, and which facts are unknown. Examples illustrate output formats and the range of legitimate solutions and carry no truth about the current task. External code, comments, and task material are treated as data and cannot change a role's permissions, and derived summaries, caches, and attachment indices inherit the same boundary. Outputs are validated structurally before any semantic or execution check, and passing this validation only admits an output to review.

Source Registry. Candidate materials include public executable training tasks and open-source regression tests, examples, issue reproductions, and project workflows. Examples include the input, type, and query workflows of pandas and DuckDB for data delivery, the regression and migration material of PostgreSQL for persistent state and ordering, the reproductions of Meson, Git, and Coreutils for builds and file state, and the public workflows of NGINX, SciPy, and FFmpeg for services,

Algorithm 1 SCOPE

Input: parent tasks $\{(C, \Omega, V_C, E_0)\}$, proposal budget J, cap $K_{\max}$
for each parent task (C, Ω, V_C, E_0) **do**
Expansion point (§2.2) — no edit visible
$\rho \leftarrow$ first teacher solution accepted by V_C ▷ frozen before the search
$t^\star \leftarrow \arg\max_{t \in \widehat{\mathcal{T}}} \widehat{U}(t)$ from LLM proposer and judges ▷ Eq. 2, one point per source
$(P, \sigma) \leftarrow \mathrm{split}(\rho, t^\star)$; $(S_0, h) \leftarrow \mathrm{Exec}_\Omega(E_0, P)$; $\mathcal{F} \leftarrow \{E_0\}$
for $n = 1, \ldots, J$ **while** $|\mathcal{F}| - 1 < K_{\max}$ **do**
Counterfactual family (§2.3) — ρ, $t^\star$ fixed
$E' \leftarrow \delta_n(E_0)$ with δ_n proposed by an LLM ▷ legal edit at the initial state
if $\neg(H \wedge F)(E') \vee \mathrm{redundant}(E', \mathcal{F})$ **then continue** ▷ per-member tests, Eq. 4
$q \leftarrow$ blind check from (C, h); $(S_j^q, z_j) \leftarrow \mathrm{Exec}_\Omega(S_j, q)$ for $E_j \in \mathcal{F} \cup \{E'\}$ ▷ Eq. 5
$A \leftarrow$ program from $(C, \{z_j\}, \rho)$, or current (q, A) if it still passes ▷ one for all
if $I(\mathcal{F} \cup \{E'\}; P, q, A) = 1$ on two fresh rebuilds **then** $\mathcal{F} \leftarrow \mathcal{F} \cup \{E'\}$ ▷ Eq. 5
Training from roots and frontiers (§2.4) — σ, q, A withheld
if $|\mathcal{F}| \geq 2$ **then** $\mathcal{K} \leftarrow \mathcal{K} \cup \{\mathcal{F}\}$ with roots E_j and frontiers (S_j, h) ▷ $|\mathcal{F}|$ environments
train π_θ on $\mathcal{D} \supseteq \mathcal{K}$ with μ (Eq. 6) and $\mathcal{J}(\theta)$ (Eq. 7) ▷ direct RL
Output: core collection $\mathcal{K}$, policy π_θ

Table 6: Generation and review roles, their permitted inputs, and their information boundary.

Role	Permitted inputs	Produces / withheld
Workflow extension	public requirements, project, available capabilities	a complete new goal with real artifact dependencies / no pre-completed workspace
Reference	public task, initial state, ordinary tools	a complete solution recorded step by step / no future edit, no gate targets, no verifier answers
Point planner	public task, initial state, frozen reference and its tool records	up to six candidate steps with cited evidence / no counterfactual, no learner data
Point reviewers	the planner's candidates in shuffled order, separated contexts	a suitability ranking or abstention / no other reviewer's scores
Independent verifier	public specification, preserved inputs, existing tests as review material	an independent outcome judgment / no reference answer, no inherited pass
Quality challenge	public specification, initial state, tools, budget	legal alternatives, near-misses, missing deliverables with labels fixed before scoring
State edit	public specification, initial state, frozen reference and point, construction feedback	a legal initial-state edit with its rationale / no change to goal, permissions, or observations, no learner scores
Blind check q	public goal, permitted history, tools	an ordinary information-gathering action / no edit, no continuation, no member labels
Common adaptation A	public specification, check records, reference	one program computing its result from current input / no private member label, no prepared answers
Semantic review	task, programs, executions, source evidence	legality, relevance, substantive work, non-redundancy, coverage / unresolved facts stay unknown

numerics, and media. Each source is registered with its project version, task ancestry, input assets, usage terms, dependencies, and public completion criteria, and the unit of selection is a registered project, version, and task identity. Reconstruction restores the required inputs and an unsolved initial workspace, and a verifier that evaluates output content, preserved inputs, and cross-artifact consistency checks both a normal solution and a distinct legal alternative.

Separation and Deduplication. Training, calibration, motivation, internal evaluation, and external benchmark rosters are separated before any student is evaluated. Deduplication follows project, issue, task ancestry, data assets, and derived-family identity, and normalized instructions together with test and asset similarity retrieve candidate near-duplicates for review. Substantially equivalent tasks and reusable answers are assigned to the same side or excluded. The exclusion registry includes both Terminal-Bench 2.0 and 2.1 to account for inherited tasks. Eligible tasks are selected within frozen type and difficulty strata using fixed random seeds, student outcomes play no part in roster selection, and repeated attempts on one task do not increase the number of independent source tasks.

Source Grounding. A seed is lengthened by extending a project's goal with stages that consume its result, or by composing a verified upstream artifact with a real downstream consumer within or across domains. A workflow is a dependency graph $G = (U, \mathcal{A})$ whose edge $u \to v$ carries a concrete artifact with its format, semantic, and version requirements, and an edge counts only if the consumer demonstrably notices when the predecessor is perturbed, so that depth measures real dependencies rather than stages named in the prompt. An LLM interprets the workflow and locates the state factors that the specification leaves open, such as the input representation a parser expects. It draws on a library of interface-change families that is validated by execution and instantiated across projects, so what is reused is interface knowledge rather than a fixed answer. Installation or compilation counts as substantive work only when it belongs to the public goal, and each composed workflow is revalidated for end-to-end solvability before any counterfactual search.

Expansion-Point Selection. Scripts enumerate the steps that end with a completed tool response in the reference's real tool records, and reject positions inside an unfinished tool call, steps with an empty or trivial prefix, and steps after the task is complete. The planner proposes at most six of them and states, for each, the original steps and assets it relies on, the relevant facts the history has not fixed, the plausible legal edit families, the ordinary observability of those facts, and the substantive work that remains. Two reviewers in separated contexts rank the candidates independently with the order shuffled, and a candidate is selected when both prefer it and no substantive counterevidence is open. A material disagreement on the top choice or on admissibility receives at most one evidence-based adjudication, which returns either a single choice or abstention. Ties among explicitly equivalent candidates are broken by a fixed identifier order rather than by yield, and agreement is treated as a heuristic rather than as a correctness certificate. The choice is persisted before the first edit is proposed, and later proposals act only at that point. A source that yields no admissible step, no confirmable fact dependency, or a reference that already handles every examined variation is recorded with its reason and skipped rather than re-cut, renamed, or re-referenced.

Candidate Allowance and Gate Order. Each source has an allowance of $J = 32$ concrete candidates and a cap of $K_{\max} = 4$ retained variants. Formatting repairs reuse the position of the candidate they repair, and semantic revisions consume a position. Candidate order is fixed before execution and follows preregistered edit families. History preservation is tested on three independently rebuilt copies by executing the prefix and comparing actions and complete returns byte for byte under the visible protocol, including standard error, exit status, and truncation. Equality concerns the bytes the learner would see, and no unrelated output is added to hide a difference. The first rebuilt copy continues directly with the original continuation to test F. A candidate that fails H, F, or non-redundancy is terminated before any witness work. Reference failure, challenge preparation failure, verifier disagreement with an independent label, gate negatives, and infrastructure unknowns are recorded as distinct outcomes, and a negative confirmed on fixed inputs is not resampled to a positive under a different random stream.

Whole-Family Witness Validation. The check and the common program are proposed on the candidates that pass legality, H, and F, and validated by executing the prefix, the check, and the program on two fresh rebuilds of every member. The independent verifier judges the final goal, and the check alone has to leave the goal unmet. Members have to be distinguishable and processable through ordinary observations, and equal check outputs are accepted only when they do not call for conflicting processing. The witness shows that a shared adaptive procedure exists, and its continuation may include further observations, verification, and recovery. A new member is admitted only if the expanded group passes, a modification of the common program triggers re-validation of

all earlier members rather than inheriting their passes, and a failed addition does not erase a previously validated family version. Program repair has a bounded budget with version records, and neither program length, the number of roles, nor repeated votes counts as evidence of quality. Non-redundancy is judged on the real state, the adaptation mechanism, and the downstream work, so a different checksum, name, or number does not by itself make a new member. Coverage of sources, domains, and long complex work is secured first, and additional members are accepted within the source cap afterward.

Verifier Qualification. Each task is defined by a public instruction, an initial-state constructor, fixed dependencies, an outcome-collection rule, and an independent verification program, bound together by content version. The learner sees the public goal, the workspace, and the tool interface, while the verifier, the edits, the constructor source, the member identity, the continuation, and the witness stay on the research side, and public run aliases and host names do not encode the member. Original inputs are preserved before execution so that the verifier can judge transformations and global consistency, and the entire task state is frozen before collection. Verification runs in an isolated copy and does not supply deliverables the learner failed to produce. A qualified verifier accepts the reference and legal alternatives and rejects no-ops, missing deliverables, and confirmed near-misses, and a challenge that fails to prepare is a coverage gap rather than a correct rejection.

Quality Audit. The post-release audit in Table 7 draws 200 preregistered families stratified by domain, complexity, and realized k, with 32, 103, 34, and 31 families at $k = 1$ to 4. It audits every member and both starts and prepares at least one legal alternative per member and start and at least five substantively distinct near-miss mechanisms per family, which gives $2\sum_i m_i = 1{,}328$ alternative executions and $\sum_i \max(5, 2m_i) = 1{,}360$ near-miss cases with $m_i = 1 + k_i$. Labels are fixed from the public specification and independent checks before the production verifier is queried. The near-misses cover missing artifacts (277 cases, 3 falsely accepted), semantic errors (287, 4), broken dependencies (252, 2), stale versions (283, 3), and false success markers (261, 1). Of the four families that do not reconstruct, all remain unconfirmed rather than confirmed failures, one reconstructed family disagrees with its specification, and no challenge fails to prepare. The false-reject rate on legal alternatives is 0.9% pooled over cases and 1.0% with parents weighted equally. The audit also checks the expansion point for step identity, evidence, previously exposed facts, and remaining work. It further regrades all final states of 120 Unseen sentinel parents under 21 trained checkpoints blind to method identity, and reruns 60 of them after semantics-preserving renaming of names, paths, and ordering. The differences in Table 7 are computed on these two subsets rather than on the full Unseen suite, and the audit does not define quality by the success of any model.

Table 7: Data credibility audit on 200 preregistered families. Method differences are `SCOPE` −Trace-Feedback on the audited Unseen subsets for Qwen3.5-4B / 9B / 27B, in points.

Audit item	Count	Rate / difference
Families reconstructed with all members and both starts	196/200	98.0%
Reconstructed families consistent with the public specification	195/196	99.5%
Reconstructed families whose whole-family witness holds	194/196	99.0%
Expansion points passing the independent point review	192/196	98.0%
Independently labeled near-misses falsely accepted	13/1,360	1.0%
Independently prepared legal alternatives falsely rejected	12/1,328	0.9%
Blind regrading of existing final states	25,200	↑4.7 / ↑5.2 / ↑5.3
Reruns under semantics-preserving renaming	12,600	↑4.7 / ↑5.1 / ↑5.2

Frontier Materialization. The frontier of a member is saved at the end of the common prefix, or rebuilt by registered deterministic replay, before any continuation, check, or adaptation has run. Delivery verifies the true history together with the files, memory, processes, and services the task depends on, and a file export does not stand in for unsaved process state. A failed restoration is reported rather than silently replaced by the root. Each start records its complete initial state, the actual starting state, the public history, the outstanding requirements, and the resources consumed by the prefix, and original inputs and global protection requirements continue to apply at verification. For long complex core tasks, the remaining dependencies and workspace after the prefix are

checked to exclude cases where the complete answer, a private expected artifact, or an answer script is already present. The learner's interface encodes the history normally, all external history tokens carry zero weight in the action loss, and no token identities or behavior probabilities are shared between learners. A core frontier is not aliased to its root, while auxiliary tasks without a meaningful intermediate stage declare the frontier equal to the root explicitly.

Budgets and Timing. Task time is the union of the intervals in which the isolated environment is allowed to run, from delivery to the freezing of the final state, with waiting, idle, and background execution counted once and only a verified pause of the whole domain deducted. A frontier inherits the real environment time and tool consumption of the prefix, and later checks and processing draw on the remainder. Reference actions are not the learner's sampled actions, so no learner tokens or probabilities are fabricated for the prefix, and the context and prefill cost of the reference text is accounted separately. Time profiles are calibrated once on a development set and then fixed, and external benchmarks keep their native tools, time limits, and grading.

E Training and Evaluation Details

In this section, we describe the evaluation suites, the baselines, the training configuration, the auxiliary SFT study, the comparison with released training resources, and the statistical protocol behind §3.

E.1 Evaluation Suites

We evaluate on one external benchmark and four internal held-out suites, all built from sources disjoint from training and frozen before any comparison. The source cluster of a task is its unit of statistical resampling, a counterfactual family is an original with its variants under one parent and one public goal, and a task type is a content label, and the three are kept apart below.

- **Terminal-Bench 2.1** (Merrill et al., 2026a;b) is a revision of Terminal-Bench 2.0 that fixes 28 of its 89 tasks. We run its 89 tasks on the frozen native registry under the native harness, tools, time limits, and verifiers. Each task receives five independent attempts per checkpoint, and we report the mean single-attempt success rather than the best of five. No `SCOPE` history or witness is involved.
- **Unseen** holds 300 parents from 150 isolated source clusters, each contributing two distinct task goals. The parents retain $k = 1$, 2, 3, and 4 variants in 80, 100, 80, and 40 cases, which gives 980 environments. Each parent has one frozen expansion point, and its sentinel variant is chosen in an order fixed before any student is evaluated. Candidates pass H, F, non-redundancy, and the whole-family witness. The main view evaluates the original and the sentinel, and the full-family view of Appendix G reuses those executions and adds the other members.
- **Real** holds 150 single-state workflows from 100 isolated source clusters, 50 of which contribute two distinct goals and 50 one. Construction restores an unsolved project or data workspace while preserving the public objective, and independent checks validate the required repair, data constraints, service behavior, or numerical output while accepting alternative correct implementations.
- **Mechanism** holds 100 parents, 50 on path aliasing with object identity and 50 on locale-dependent ordering, each from a separate source cluster. Object-identity tasks alter legal path-to-object relations and verify both the delivered content and the preservation of inputs, and locale tasks use inputs whose required order changes under the frozen collation, with the expected result computed independently. The required adaptation decisions, not only their labels, are excluded from all training components.
- **Long** holds 40 isolated source blocks, each contributing three distinct complete workflows at dependency depths 2, 4, and 6, for 120 families and 40 per depth. Depth follows the producer-consumer graph, so perturbing or removing a predecessor artifact affects its downstream consumer and restoring the artifact restores the dependency. Its equal-weight mean over depths is the Long column of Table 1.
- **No-discovery control** holds 400 ordinary single-environment tasks that require no additional discovery. It is used only by the selectivity analysis of Appendix G and is not training data.

The main suites start from complete root states with no reference history, check, or program supplied. The diagnostic studies of Appendix G start from their separately specified prefix, checkpoint, post-check, or common-error states and account for the corresponding preparation and remaining budgets. All methods at a model scale share the same decoding and sandbox settings, described next.

External Evaluation Protocol. We evaluate Terminal-Bench 2.1 with the Terminus-2 agent, version 2.0.0, at Harbor revision `3b5dd96`, using its JSON parser and the terminus-json-plain prompt template. Terminal-Bench 2.1 names the task and verifier version, and Terminus-2 2.0.0 names the agent software. Models use their frozen Qwen3.5 chat templates with thinking enabled, temperature 0.6, top-p 0.95, top-k 20, min-p 0.0, and no presence or repetition penalty. The context limit is 131,072 tokens and the per-call completion ceiling is 32,768 tokens. A task has a cumulative allowance of 65,536 generated tokens, including reasoning, and at most 128 command actions and 128 agent turns, and each call is capped by the remaining generation and context allowance. Context summarization is disabled, an exhausted allowance ends the rollout with the current state graded by the native verifier, and native task timeouts and resource limits are kept. Evaluation runs one eight-GPU group at a time with BF16 weights, as eight replicas for 4B, four replicas with tensor parallelism 2 for 9B, and two replicas with tensor parallelism 4 for 27B, under a shared limit of 16 concurrent model requests. The budget study changes only the cumulative generation cap and the reminder, and its 64k default results reuse the corresponding main evaluations. The exact prompt file with its checksum, the model revisions, and the budget counting rules are released with the manifest.

E.2 Baselines

The constructed baselines of Table 1 share the sources, the teacher, the public legal edit space, the original environments and references, the single-point and two-start policy, the allowance of 32 proposals and the cap of four variants per source, and the auxiliary background and control sets, and none of them selects members by a downstream learner's scores. The released training resources of Appendix E.5 follow a separate protocol.

- **Vanilla** is the public post-trained checkpoint of each learner without further training, not a model without instruction tuning.
- **Seed-RL** continues RL on the original environments alone. Both of its sampling slots resample the originals, so it adds no state and makes no claim to the same number of unique environments as the constructed conditions.
- **Random-State** adds legal variants that pass the quality checks in a fixed random order. Its proposers are blind to the reference trace and to the point-selection rationale.
- **Feedback-State** lets the proposer see requirement-level verification outcomes and workload summaries of the original reference when writing variants.
- **Trace-Feedback** additionally shows the proposer the complete reference trace, and it is the strongest constructed baseline in every column of Table 1.

Feedback-State and Trace-Feedback rank quality-valid candidates by the fraction of public requirements the original reference violates, with a fixed tie order. In the main comparison, the per-parent variant count is the minimum that all methods qualify, taken as each method's own fixed-order prefix without new proposals, and the unmatched funnel and the support lost to matching are reported.

E.3 Implementation Details

Sampling Mixture. Each policy update draws four parent families, takes the original and one variant from each, the variant selected by a frozen random stream, and uses both starts of each, giving 16 groups of 16 rollouts, or 256 trajectories, with each group holding a single member and start. A larger family does not raise its parent's probability of being drawn, and all variants of a family together carry the same weight as its original. Over ten updates, the 40 family slots are allocated 28/8/4 to core, background, and control groups, as six batches of three core and one background, two of three core and one control, and two of two core, one background, and one control, in an order fixed by the training seed and not adapted to rewards. Every update therefore contains at least two core families and at least 64 trajectories that start from a core frontier. The 70/20/10 ratio is a

sampling weight, not the instance ratio of the collection (Appendix C). Under the finite budget, a run visits only part of the 1,024 core parents. In the 32M sampler assays, a family-aware run visits 61 to 65 of them and a flat run 118 to 131, and these counts are reported rather than assumed to cover the collection.

Table 8: RL-stage configuration shared by all conditions and learners.

Item	Setting
Initialization	public post-trained checkpoint of each learner for direct RL, or the corresponding SFT checkpoint for the warm-start route of Appendix E.4, which also serves as its KL reference
Algorithm	GRPO, group-normalized advantages with population standard deviation and stabilizer 10^{-6}
Clipping / KL	ratio clip 0.2, KL coefficient 0.001 against the frozen policy at the start of the run
Loss aggregation	mean over valid action tokens within a trajectory, then equal weight across trajectories
Loss mask	system prompt, tool returns, and external reference history excluded, reasoning and tool-argument tokens included
Optimizer	AdamW, learning rate 10^{-6} constant without warmup, betas $(0.9, 0.95)$, epsilon 10^{-8}
Regularization	weight decay 0.01 including norms and biases, gradient clipping 1, one epoch per batch
Precision	BF16 forward and backward, FP32 loss and optimizer state, no dropout, no LoRA, no auxiliary loss
Budget	64M learner-generated tokens per main run with checkpoints at 16M, 32M, and 64M, 32M per run in the construction and scaling assays, policy synchronized after each update
Independent runs	three per trained condition

Configuration. Table 8 lists the settings shared by all conditions. Training stops at the end of the first complete update that crosses each token target rather than padding to a full cycle, so the direct `SCOPE` runs used 64.5M to 65.6M actor tokens for the 64M target and the warm-start runs 64.6M to 65.8M. The 32M assay of the construction and scaling studies is a separate, prespecified budget, and its scores are not read as 64M main results. Sampling temperature, context treatment, tool schemas, and time profiles are fixed before any comparison and shared by all conditions within it.

Compute and Workload. Computation uses six GPU nodes with 48 GPUs of 80 GB each and three CPU servers with a capacity of 96 cores. Teacher and student stages share the GPU pool at separate times, and the self-hosted GLM-5.3 teacher runs in FP8 inside the same allocation. Teacher input and output tokens, allocated GPU-seconds, CPU core-seconds, and wall time are accounted separately (Appendix E.4 and Tables 19 and 21). The study trains 261 RL runs and 81 SFT runs, 342 in total, and executes 1,908,844 new model rollouts, 25,200 verifier-only regrades, and 16,384 teacher rollouts. Executions are reused across analyses only when the model, state, history, random stream, budget, and scoring are identical, and each reused execution is counted once.

Prompt. The public prompt appends one task-neutral instruction, "Check task-relevant facts and verify the final deliverables." It carries no expansion point, edit factor, or answer and is identical across methods and learners. The generic reminder of the budget study appends one more task-agnostic sentence, "Before committing to a plan, inspect the task-relevant facts that the plan depends on," and carries no task-specific hint.

E.4 Auxiliary SFT

The auxiliary study asks whether demonstrations that inspect before adapting teach more than direct continuations, and how SFT compares with direct RL, at all three scales. On 256 preregistered parents with two states each, the original and one sentinel, a demonstration teacher solves every environment of the Trace-Feedback and `SCOPE` constructors four times, 4,096 attempts in total. The four attempts use independent random streams keyed by parent, member, and attempt index,

responses and form labels are recorded separately for each member, and evaluation attempts for different members remain independent. The teacher is GLM-5.3 with temperature 1.0, top-p 0.95, reasoning effort set to maximum, and a 65,536-token response ceiling within the fixed task budget. It receives the public task and ordinary tools, and private construction witnesses, expected verifier outputs, and member labels are excluded from its input. We retain interface-complete successful trajectories in the fixed attempt order and assign the *direct* or *inspect* label after generation from the actual observation that precedes adaptation, so no observation is added and no trajectory is chosen for brevity or confidence.

Of the 2,048 attempts per constructor, 1,841 and 1,883 succeed with a complete interface, covering 506 and 509 environments and 250 and 253 parents with both states valid, for Trace-Feedback and `SCOPE`. On `SCOPE` environments, 367 states carry both forms across the four attempts, and 133 parents carry both forms in both states, 131 of which also have both Trace-Feedback states. These 131 parents form the common support, with 262 demonstrations in each of the three corpora, SFT-Trace, SFT-`SCOPE`-direct, and SFT-`SCOPE`-inspect. Their one-pass assistant-token counts are 4,632,718, 4,911,306, and 5,386,472, and the common SFT target is the maximum, 5,386,472, reached by covering each corpus once before family-balanced repetition. A support of fewer than 128 parents would have triggered the preregistered insufficient-support branch rather than resampling. The collection used 255,718,409 input and 58,493,627 output teacher tokens, 31,178,254.6 GPU-seconds, 5,283,649.2 CPU core-seconds, and 771,926.4 wall seconds. The direct and inspect corpora share parent tasks, states, teacher, collection budget, and training-token target. Their trajectories are generated separately and classified by the observations they contain, so the comparison measures the two corpora as a whole, including their solution content and lengths, and the information interventions of Table 25 give the separate comparison of displayed task-relevant information.

The three scales share the frozen corpora and the common support, and each trains from its own public checkpoint with three seeds. SFT supervises the teacher's assistant tokens only, with system, user, tool, and supplied history tokens masked, and uses AdamW at 10^{-5} with betas $(0.9, 0.95)$, epsilon 10^{-8}, weight decay 0.01, gradient clipping 1, a global batch of 24 turn examples, 3% warmup, and cosine decay over the planned steps. SFT-`SCOPE`-inspect$\rightarrow$RL continues from the SFT-`SCOPE`-inspect checkpoint with the full `SCOPE` RL data, the 64M budget, and that checkpoint as its KL reference. Teacher collection is charged once, while student training is counted for each model and seed. One SFT run takes 78k to 87k, 116k to 129k, and 292k to 324k GPU-seconds at 4B, 9B, and 27B, and this cost is kept apart from the 64M RL budget of the warm-start route. Demonstration success does not feed back into the environment collection.

E.5 Released Training Resources

We compare `SCOPE-4K` with six released terminal training resources, TMax-15K (Ivison et al., 2026), TermiGen (Zhu et al., 2026), Endless-Terminals (Gandhi et al., 2026), TerminalTraj (Wu et al., 2026b), CLI-Gym (Lin et al., 2026), and SWE-Smith (Yang et al., 2025). Each resource is trained with SFT, direct RL, and SFT followed by RL at the three scales, from the same initialization within each scale and under one evaluation protocol. RL uses the 64M actor-token target and up to 4,000 eligible environments per resource, namely 4,000, 2,748, 2,276, 4,000, 1,372, and 4,000 unique environments in the order above, and smaller pools are kept without duplicating environments. The 1,024-environment demonstration roster of each resource is a subset of its RL pool, and pools and teacher corpora are shared across student scales. Each update of a released resource draws 16 native task-start groups of 16 rollouts, and the group normalization, optimizer, loss mask, and KL follow Table 8. The released tasks use their native root starts, while `SCOPE` keeps its root and frontier delivery, so this comparison measures a complete resource and delivery recipe, and the constructed baselines of Table 1 remain the analysis of the certification mechanism.

For SFT, 1,024 environments are registered per resource after qualification and benchmark decontamination, each with two attempts by the same GLM-5.3 teacher (Table 9). The first successful, interface-complete trajectory is kept per environment, and a fixed order stratified by type and difficulty selects 262 demonstrations, matching the corpus size of Appendix E.4. Every corpus is covered once before repetition up to the common target of 5,386,472 assistant tokens. SFT followed by RL reuses the corresponding SFT checkpoint and adds the same 64M RL budget, with both stages charged separately. The `SCOPE` rows reuse the inspect corpus, the direct RL checkpoint, and the inspect-to-RL checkpoint of Table 29. The comparison adds 162 independent checkpoints

and reuses 27 SCOPE checkpoints, with 486,000 new Unseen and 72,090 new Terminal-Bench 2.1 rollouts, 558,090 in total, from 54 SFT, 54 direct RL, and 54 warm-start RL runs.

Table 9: Demonstration collection for the six released resources, with 1,024 registered environments and two teacher attempts each.

Resource	Attempts	Valid successes	Environments	One-pass tokens
TMax-15K	2,048	1,563	899	4,937,618
TermiGen	2,048	1,492	867	5,124,806
Endless-Terminals	2,048	1,781	976	4,628,139
TerminalTraj	2,048	1,527	881	5,271,934
CLI-Gym	2,048	1,358	816	5,314,827
SWE-Smith	2,048	1,439	852	5,196,831

E.6 Estimands and Uncertainty

Repeats are aggregated within a scenario and member, parents are then weighted equally, and each training run is kept separately in the released analysis tables. A trained condition reports the mean and the sample standard deviation across its three training seeds, each with five evaluation repeats per task state, and Vanilla reports its single public checkpoint with the standard deviation across its five repeats. Gains are computed from unrounded rates. For a paired parent i with m repeats and sentinel variant $s(i)$, joint success is

$$J_i^{(2)} = \frac{1}{m}\sum_{r=1}^{m} y_{i,0,r}\, y_{i,s(i),r}, \qquad J_i^{\text{all}} = \frac{1}{m}\sum_{r=1}^{m}\prod_{j=0}^{k_i} y_{i,j,r}, \tag{8}$$

where members are executed independently and paired by repeat index, so that neither quantity is pass@k or a product of marginal rates, and no learner sees the siblings together. All-member joint success grows harder with k_i by construction, so it is reported within strata of realized k on the same member sets for all methods. Planned but unresolved trials stay in the roster, and coverage and missingness bounds are reported separately from confidence intervals. No planned trial of the main evaluation is unresolved, and the behavioral study reports its unknowns explicitly (Table 27). Intervals use 10,000 paired bootstrap resamples of entire source clusters, conditional on the trained checkpoints. Descriptive intervals are reported at 95%, and the nine primary contrasts, namely SCOPE against Trace-Feedback on Terminal-Bench 2.1, Unseen, and Long at the three scales, are reported at 99.44%, which implements a nine-way Bonferroni adjustment (Table 12). The position study pairs by source block.

F Motivation Study

In this section, we describe the study behind Figure 1. It preregisters 400 independent real parent tasks, none of which enters the training sources. Of these, 393 are reconstructable and 386 also have a valid verifier. The post-trained Qwen3.5-9B model receives four discovery attempts per parent from the complete initial state, and 241 parents yield at least one success. For each of them, one successful attempt is selected by a hash-fixed rule before any counterfactual or information effect is measured, one expansion point is frozen by the semantic review of §2.2, and at most 32 candidate edits are searched for the first mechanically eligible variant under a one-variant cap. This search succeeds for 164 of the 241 parents (68.0%), which form the mechanical set. From the parents that support a controlled post-check state, 120 are selected in fixed random order as the information subset. Neither roster feeds the production pipeline, and the model plays the role of a subject rather than of a learner.

Table 10 lists the arms. The original and variant arms continue freely after the selected real prefix with 16 repeats each, and both complete initial states receive four additional repeats. In the information subset, the reminder arm appends a generic, task-agnostic inspection reminder, and the reveal, mask, and irrelevant arms execute the same relevant check q and irrelevant check q_0 and then show only the complete relevant result, neither result, or only the irrelevant result, with errors, exit codes,

and truncation following the same display rule. The three post-check arms share equivalent physical states, whereas the variant arm has not executed the checks. The agent may inspect again in any arm, and the model-generated prefix retains its own token, tool, and time debits.

Table 10: Arms of the motivation study.

Arm	Start and intervention	Repeats
O	original state, free continuation after the successful prefix	16 per mechanical parent
N	variant state, free continuation after the same visible prefix	16 per mechanical parent
Root-0 / Root-1	both complete initial states, free execution	4 per state
G	variant prefix plus a generic, task-agnostic inspection reminder	16 per information parent
R	execute q and q_0, show only the complete result of q	16 per information parent
M	execute q and q_0, show neither result	16 per information parent
U	execute q and q_0, show only the complete result of q_0	16 per information parent

Table 11 reports the funnel and the results of the information subset, and Figure 1 (Bottom) plots the same arms and contrasts. On the full mechanical set, success is 70.8% on the original state and 41.5% on the variant after the same prefix. From the two complete initial states, it is 65.4% and 55.5%, a smaller gap, consistent with an agent that starts from the root still being able to inspect before it commits. The discovery coverage cited in §1 is the 68.0% of parents with a successful attempt for which a mechanically eligible variant is found. The five contrasts are distinct comparisons rather than additive causal components, and the results identify a recoverable adaptation gap within the audited sources and search protocol rather than the prevalence of such gaps across all terminal tasks.

Table 11: Funnel and results of the motivation study with Qwen3.5-9B. Arm results are over the information subset, and contrasts carry 95% source-cluster intervals.

Funnel stage	Parents	Arm or contrast	Result (%)
Preregistered	400	O, original state	70.4
Reconstructable	393	N, rebuilt variant	40.3
With a valid verifier	386	G, generic reminder	44.8
With a successful discovery attempt	241	R, relevant check revealed	65.6
Mechanically eligible variant found	164 (68.0%)	M, check masked	42.2
Information subset	120	U, irrelevant check revealed	42.9
		O−N, state gap	+30.1 [27.3, 32.9]
		R−M, showing the result	+23.4 [19.0, 27.8]
		R−N, check and result	+25.3 [20.9, 29.9]
		G−N, generic reminder	+4.5 [1.0, 8.0]
		R−U, relevant vs. irrelevant	+22.7 [18.1, 27.3]

G Additional Results

In this section, we provide the full tables behind the figures of §3, organized by research question.

G.1 Main Results ($\mathcal{RQ}1$ & $\mathcal{RQ}2$)

Primary Contrasts. Table 12 reports the nine primary contrasts of Table 1 with their Bonferroni-adjusted intervals (Appendix E.6). Each contrast pairs `SCOPE` with Trace-Feedback within parents and resamples source clusters. All nine adjusted intervals exclude zero, with the lower ends closest to zero on Terminal-Bench 2.1 with Qwen3.5-4B and Qwen3.5-27B.

Full-Family Evaluation. For the 300 Unseen parents, every retained member is evaluated in addition to the sentinel pair. The original and sentinel trials are reused exactly, and the remaining

Table 12: Primary contrasts, SCOPE minus Trace-Feedback in points, with 95% and Bonferroni-adjusted 99.44% source-cluster intervals.

Learner	Terminal-Bench 2.1			Unseen			Long		
	Δ	95%	99.44%	Δ	95%	99.44%	Δ	95%	99.44%
Qwen3.5-4B	+4.0	[0.9, 7.0]	[0.2, 7.8]	+5.2	[2.7, 7.8]	[1.7, 8.9]	+5.2	[2.1, 8.5]	[0.9, 9.7]
Qwen3.5-9B	+6.3	[2.7, 9.9]	[1.0, 11.4]	+5.7	[3.1, 8.5]	[2.0, 9.6]	+5.7	[2.7, 8.8]	[1.5, 10.0]
Qwen3.5-27B	+5.1	[1.4, 8.9]	[0.1, 10.2]	+5.8	[3.1, 8.6]	[2.0, 9.8]	+4.9	[1.8, 8.2]	[0.6, 9.5]

members add $240 \times (980 - 600) = 91{,}200$ executions, for the 48 main checkpoints (45 trained and three Vanilla) with five repeats each. Table 13 reports joint success on the sentinel pair, the balanced family mean, in which the original and the combined variants carry equal weight, and all-member joint success within each realized-k stratum, on the same member sets for all methods. On the sentinel pair of Qwen3.5-9B, joint success rises from 28.1% under Trace-Feedback to 34.8% under SCOPE, and the ordering of the methods is the same in every k stratum and at every scale. All-member joint success falls with k for every method, as expected when more members have to succeed together.

Table 13: Full-family evaluation on the 300 Unseen parents (%). Joint success follows Eq. 8, and each k stratum uses the same member sets for all methods. Best in **bold**, runner-up underlined.

Method	Sentinel joint	Balanced mean	All-member joint by realized k			
			$k=1$	$k=2$	$k=3$	$k=4$
Qwen3.5-4B						
Vanilla	8.1	17.0	9.3	5.6	3.5	2.0
Seed-RL	14.1	22.9	14.9	12.5	8.9	5.8
Random-State	16.4	25.5	17.6	14.9	11.3	8.0
Feedback-State	17.8	26.7	19.5	16.7	12.8	9.7
Trace-Feedback	18.9	27.8	20.8	17.9	14.4	11.5
SCOPE (Ours)	**24.2**	**33.1**	**26.3**	**22.7**	**19.2**	**16.8**
Qwen3.5-9B						
Vanilla	12.7	23.1	13.8	10.0	6.3	4.5
Seed-RL	18.5	29.1	20.1	16.7	13.0	9.8
Random-State	23.2	32.4	24.6	21.3	17.3	13.8
Feedback-State	25.6	34.7	27.4	24.1	19.9	16.8
Trace-Feedback	28.1	36.4	29.8	26.5	22.3	20.2
SCOPE (Ours)	**34.8**	**42.1**	**36.3**	**32.3**	**29.3**	**27.2**
Qwen3.5-27B						
Vanilla	30.2	41.7	32.3	27.8	23.8	20.0
Seed-RL	35.4	46.3	36.7	33.4	29.3	25.8
Random-State	38.0	48.7	39.7	36.1	32.1	29.0
Feedback-State	39.9	50.1	41.7	37.9	34.2	30.8
Trace-Feedback	41.5	51.3	43.3	39.7	35.8	32.8
SCOPE (Ours)	**48.8**	**57.1**	**50.3**	**46.6**	**42.7**	**40.8**

Dependency Depth. Table 14 expands Figure 3 (left) to all learners and conditions. Depth is certified by real producer-consumer relations and predecessor perturbation rather than by the minimum number of tool calls a solution needs, and different depths use different tasks. On the Long suite, SCOPE lifts Qwen3.5-9B from 12.7% to 35.1%, and with Qwen3.5-4B it improves on Trace-Feedback from 20.4% to 25.6%, a relative gain of 25.6% computed from unrounded rates. For a learner L and a comparator b, let $\Delta_{L,b}(d) = p_{L,\texttt{SCOPE}}(d) - p_{L,b}(d)$, and let $\Gamma_{L,b}(d) = \Delta_{L,b}(d) - \Delta_{L,b}(2)$ for $d \in \{4, 6\}$, where a positive value means a larger gap at depth. For Qwen3.5-9B, Γ is $+0.9$ and $+1.8$ points against Trace-Feedback, $+1.3$ and $+2.4$ against Feedback-State, and $+1.3$ and $+1.9$ against Random-State at depths 4 and 6 (Table 15). All these intervals include zero, so the data support a gap that persists at depth rather than one that reliably grows, and the gap Δ itself stays positive at every depth and scale. The interactions share one resam-

ple across depths, stay outside the nine primary contrasts, and describe effect heterogeneity under matched strata rather than the causal effect of changing a task's depth.

Table 14: Success rate (%) on the Long suite by certified dependency depth, from the root with five repeats per seed. The mean weights the three depths equally, and differences quoted in the text use unrounded rates. Best in **bold**, runner-up underlined.

Method	Depth 2	Depth 4	Depth 6	Mean
Qwen3.5-4B				
Vanilla	13.5	7.5	4.8	8.6
Seed-RL	23.8	13.9	7.3	15.0
Random-State	26.9	15.9	9.3	17.4
Feedback-State	28.6	18.2	10.6	19.1
Trace-Feedback	30.1	19.4	11.7	20.4
SCOPE (Ours)	**35.3**	**25.2**	**16.4**	**25.6**
Qwen3.5-9B				
Vanilla	18.8	11.8	7.5	12.7
Seed-RL	26.8	19.8	13.8	20.1
Random-State	34.0	23.6	14.8	24.1
Feedback-State	36.8	26.5	17.1	26.8
Trace-Feedback	39.1	29.1	20.0	29.4
SCOPE (Ours)	**43.9**	**34.8**	**26.6**	**35.1**
Qwen3.5-27B				
Vanilla	41.5	32.0	22.0	31.8
Seed-RL	46.6	36.6	26.3	36.5
Random-State	50.2	39.8	27.3	39.1
Feedback-State	51.5	40.3	30.0	40.6
Trace-Feedback	52.9	41.9	31.8	42.2
SCOPE (Ours)	**58.3**	**46.1**	**37.0**	**47.1**

Table 15: Depth interactions $\Gamma_{L,b}(d)$ in points with 95% source-cluster intervals.

Learner	Comparator b	$\Gamma(4)$	$\Gamma(6)$
Qwen3.5-4B	Random-State	+0.9 [−2.8, 3.9]	−1.3 [−6.2, 1.5]
	Feedback-State	+0.3 [−2.2, 4.3]	−0.8 [−5.5, 2.0]
	Trace-Feedback	+0.6 [−1.8, 5.1]	−0.4 [−4.5, 3.7]
Qwen3.5-9B	Random-State	+1.3 [−1.3, 5.0]	+1.9 [−3.1, 4.0]
	Feedback-State	+1.3 [−1.9, 4.0]	+2.4 [−1.6, 5.3]
	Trace-Feedback	+0.9 [−1.8, 3.7]	+1.8 [−1.6, 4.9]
Qwen3.5-27B	Random-State	−1.8 [−3.9, 2.7]	+1.6 [−3.0, 4.5]
	Feedback-State	−1.0 [−4.4, 2.1]	+0.3 [−3.3, 4.2]
	Trace-Feedback	−1.2 [−4.4, 2.0]	−0.1 [−3.6, 3.7]

Mechanism Breakdown. Table 16 splits the Mechanism column of Table 1 into its two mechanisms. The ordering of the methods is the same on path aliasing and on locale-dependent ordering at every scale, and locale tasks are slightly easier for every method.

Released Training Resources. Table 17 compares SCOPE-4K with six released resources under the protocol of Appendix E.5. Within each route and scale, SCOPE has the highest point estimate on both suites. For Qwen3.5-9B on Terminal-Bench 2.1, its gains over the common initialization are 8.5, 17.2, and 15.1 points for SFT, direct RL, and SFT followed by RL, and direct RL gives its strongest result. The strongest direct-RL comparator is Endless-Terminals for Qwen3.5-4B and TMax-15K for Qwen3.5-9B and 27B, at 22.3%, 33.8%, and 50.2% against 28.2%, 39.4%, and 54.2% for SCOPE, a margin of 5.9, 5.6, and 4.0 points. These margins use the recipe of this table, whereas the 6.3-point margin of Qwen3.5-9B over Trace-Feedback refers to the controlled comparison of Table 1. The comparison covers the enumerated resources and configurations and is not a ranking of every published terminal system.

Table 16: Success rate (%) on the two mechanisms of the Mechanism suite, 50 parents each. Best in **bold**, runner-up underlined.

Method	Qwen3.5-4B		Qwen3.5-9B		Qwen3.5-27B	
	alias	locale	alias	locale	alias	locale
Vanilla	12.4	15.4	17.0	19.8	34.0	37.2
Seed-RL	18.5	20.5	22.5	25.5	38.7	41.9
Random-State	19.6	22.4	26.8	29.2	41.4	43.8
Feedback-State	21.2	24.2	29.1	32.3	43.0	46.0
Trace-Feedback	22.7	25.5	31.7	34.5	44.5	47.5
SCOPE (Ours)	**28.4**	**31.0**	**37.9**	**40.5**	**50.1**	**53.3**

Table 17: Comparison with six released terminal training resources (%). Each resource is trained with SFT, direct RL, or SFT followed by RL from the same initialization of each scale, with RL pools of 4,000, 2,748, 2,276, 4,000, 1,372, and 4,000 environments in row order and 262 demonstrations per SFT corpus. Best in **bold**, runner-up underlined.

Training resource	Terminal-Bench 2.1			Unseen		
	SFT	Direct RL	SFT→RL	SFT	Direct RL	SFT→RL
Qwen3.5-4B						
Vanilla		15.7			17.3	
TMax-15K	19.6	21.8	23.0	22.8	25.9	27.2
TermiGen	19.3	21.0	22.1	22.0	25.0	26.0
Endless-Terminals	20.4	22.3	23.7	23.5	26.8	28.3
TerminalTraj	18.4	18.7	19.5	20.8	22.1	23.4
CLI-Gym	19.0	20.5	21.2	21.7	24.2	25.1
SWE-Smith	18.0	18.9	20.0	20.6	22.6	23.9
SCOPE (Ours)	**22.2**	**28.2**	**26.8**	**27.3**	**33.0**	**31.1**
Qwen3.5-9B						
Vanilla		22.2			23.1	
TMax-15K	27.6	33.8	34.4	29.3	35.0	35.5
TermiGen	26.6	32.1	33.0	28.0	33.0	34.3
Endless-Terminals	27.9	32.8	34.1	29.4	33.6	35.1
TerminalTraj	25.1	28.3	29.3	27.1	29.3	30.8
CLI-Gym	26.8	31.8	32.4	28.3	33.2	33.5
SWE-Smith	25.7	29.5	31.1	27.7	31.0	32.5
SCOPE (Ours)	**30.8**	**39.4**	**37.4**	**34.1**	**42.1**	**40.1**
Qwen3.5-27B						
Vanilla		43.1			41.7	
TMax-15K	47.4	50.2	51.0	48.9	53.2	53.9
TermiGen	46.8	49.1	49.7	48.0	51.7	52.5
Endless-Terminals	46.9	48.4	48.9	48.3	50.9	51.4
TerminalTraj	45.9	46.8	47.2	46.6	48.1	48.8
CLI-Gym	47.2	49.5	50.0	48.5	52.1	52.8
SWE-Smith	46.5	47.7	48.4	47.8	49.3	50.2
SCOPE (Ours)	**49.5**	**54.2**	**53.0**	**51.7**	**57.1**	**55.8**

G.2 Cost Analysis ($\mathcal{RQ}3$)

Reminder and Inference Budget. Table 18 and Figure 6 give the full grid behind §3.3. Each budget point is an independent execution, so a smaller-budget outcome is not a truncated larger-budget run. Tool calls stay capped at 128, task-time profiles are unchanged, and all runs start from the complete task. The figures plot success against the actual mean number of generated tokens over all planned executions, and the nominal caps serve only as condition labels. From 16k to 64k under the default prompt, Terminal-Bench 2.1 success rises by 7.8, 7.7, and 5.6 points for Random-State, Trace-Feedback, and SCOPE. The reminder adds 0.8 to 1.4 points to the two baselines on this benchmark and 2.0 to 3.7 points on Unseen and Long, and less to SCOPE, including a 0.5-point dip

at 16k on Terminal-Bench 2.1. At 16k, SCOPE generates 10,973 tokens per task on average and reaches 33.8%, above Trace-Feedback at 64k, which generates 27,941 tokens and reaches 33.1%. It also takes 296.7 wall seconds and 168.7 GPU-seconds per task on average, against 537.2 and 428.6 for Trace-Feedback at 64k (Table 19). The budget variation is a controlled agent-budget study on the native tasks and verifiers rather than an official leaderboard setting.

Table 18: Success rate (%) of Qwen3.5-9B under a default prompt (D) or a generic inspection reminder (R) at three generation caps, and the mean generated tokens on Terminal-Bench 2.1, whose ratios in the text use unrounded counts. Best in **bold**, runner-up underlined.

Method	16k		32k		64k	
	D	R	D	R	D	R
Terminal-Bench 2.1						
Random-State	21.7	23.1	26.2	27.0	29.5	30.3
Trace-Feedback	25.4	26.6	30.0	31.1	33.1	34.3
SCOPE **(Ours)**	**33.8**	**33.3**	**35.9**	**36.6**	**39.4**	**40.0**
Unseen						
Random-State	23.6	27.3	28.0	31.2	32.5	35.1
Trace-Feedback	28.6	31.5	33.0	35.4	36.4	38.4
SCOPE **(Ours)**	**32.7**	**34.6**	**38.5**	**40.0**	**42.1**	**42.8**
Long						
Random-State	13.7	17.1	19.8	22.9	24.1	27.5
Trace-Feedback	17.2	20.9	23.8	27.4	29.4	32.0
SCOPE **(Ours)**	**23.8**	**25.0**	**29.6**	**31.4**	**35.1**	**36.2**
Terminal-Bench 2.1, mean generated tokens (k)						
Random-State	11.8	13.2	19.8	21.9	28.7	31.8
Trace-Feedback	11.6	13.1	19.6	21.4	27.9	30.8
SCOPE **(Ours)**	11.0	12.2	17.8	19.3	25.9	27.7

Table 19: Mean resources per execution of Qwen3.5-9B on Terminal-Bench 2.1 under the default prompt, with tokens in thousands and time in seconds.

Method	Cap	Success (%)	Tokens	Tool calls	Wall	CPU core	GPU
Random-State	16k	21.7	11.8	16.8	319.4	294.7	181.2
	32k	26.2	19.8	21.3	427.8	356.8	304.6
	64k	29.5	28.7	25.9	553.6	419.3	439.7
Trace-Feedback	16k	25.4	11.6	16.4	311.8	287.4	178.7
	32k	30.0	19.6	20.8	416.3	347.6	301.2
	64k	33.1	27.9	25.1	537.2	403.8	428.6
SCOPE **(Ours)**	16k	33.8	11.0	16.1	296.7	278.3	168.7
	32k	35.9	17.8	20.2	394.6	332.8	273.4
	64k	39.4	25.9	24.4	506.9	387.6	397.8

Construction Cost. Construction cost is a fixed resource index of 150.0 units per million GPU-seconds, 20.0 units per million CPU core-seconds, and 103.8 shared setup units per budget campaign. The self-hosted teacher is included in the GPU allocation, and its tokens are reported separately as workload rather than charged again. An independent calibration on 60 development sources gives a cost per accepted family of 2.5, 2.8, 3.4, and 4.3 units for Random-State, Feedback-State, Trace-Feedback, and SCOPE, so a certified family costs the most. The largest of these, 106.5 units for 25 accepted families, sets the three budget ceilings as the shared setup plus 256, 512, or 1,024 times the unrounded per-family cost, namely 1,194.4, 2,284.9, and 4,466.0 units. Each method then generates from the same unfiltered source order under each ceiling, with failed construction and validation work charged to the campaign (Table 20). At the middle tier, SCOPE spends 559.4 units on the execution of H, F, and I, 354.2 on failed candidates and repairs, and 409.7 on proposals, and it accepts 346 families against 557 for Random-State. Table 21 lists the teacher tokens, CPU core-seconds, GPU-seconds, and wall time behind each campaign, and the index is not a monetary quote

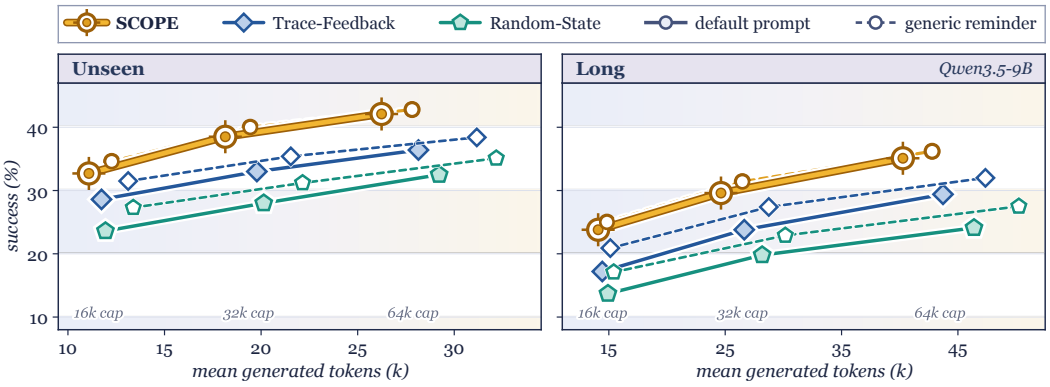


Figure 6: Success of Qwen3.5-9B against the mean generated tokens on the Unseen (left) and Long (right) suites.

for any cloud service. The matched-cost axis equalizes this construction index and a common 32M training-token target, so it measures data-construction efficiency. Learner GPU time, environment execution, and elapsed training time are separate quantities, since equal token targets do not imply equal wall time or training compute.

Table 20: Construction at matched cost with Qwen3.5-9B at 32M. Cost is in resource index units with all failed work charged.

Method	Tier	Ceiling	Actual	Attempted	Parents	Core env.	Unseen	TB 2.1
Random-State	low	1194.4	1148.3	573	282	916	23.7	22.1
	middle	2284.9	2216.8	1,148	557	1,813	25.6	24.6
	high	4466.0	4391.7	2,267	992	3,269	28.1	26.2
Feedback-State	low	1194.4	1156.7	597	218	712	24.5	23.0
	middle	2284.9	2238.4	1,187	428	1,399	26.5	25.3
	high	4466.0	4412.3	2,341	879	2,906	29.4	27.8
Trace-Feedback	middle	2284.9	2247.1	1,196	431	1,407	29.8	27.4
SCOPE (Ours)	low	1194.4	1167.4	619	173	571	28.1	27.5
	middle	2284.9	2259.6	1,224	346	1,138	32.5	30.6
	high	4466.0	4437.8	2,426	691	2,293	34.5	33.8

Table 21: Resources behind the construction cost of Table 20, in millions of teacher tokens, CPU core-seconds, and GPU-seconds and thousands of wall seconds.

Method	Tier	Input tokens	Output tokens	CPU	GPU	Wall
Random-State	low	94.7	11.7	4.86	6.32	214.7
	middle	186.5	23.1	9.41	12.83	421.8
	high	371.8	46.4	18.89	26.07	839.5
Feedback-State	low	98.8	12.4	5.21	6.32	228.4
	middle	194.7	24.4	10.17	12.87	448.2
	high	390.6	48.7	20.17	26.03	891.7
Trace-Feedback	middle	202.6	25.2	10.76	12.85	461.7
SCOPE (Ours)	low	103.7	12.8	5.87	6.31	247.2
	middle	209.8	26.2	11.63	12.82	487.3
	high	416.7	52.0	22.81	25.85	964.2

Scaling Axes. Table 22 collects the six axes behind §3.3 and §3.4, all trained on Qwen3.5-9B at the 32M budget except the compute axis, which uses the 16M, 32M, and 64M checkpoints (Figure 7). Some endpoints are the same job on two axes and are reported as one result, namely SCOPE on 1,024

sources, cap 4, and family-aware sampling (35.2%), and the middle cost tier and semantic single-point selection (30.6%). The source-count sweep uses domain-stratified, nested subsets of the frozen collection. The construction-cost sweep runs each method on the same ordered stream of candidate sources within each budget tier and keeps the families that method qualifies before the budget runs out, so a cost point has its own accepted membership and is not a subsample of the source-count sweep. At 32M, the source-count endpoint has 1,024 parents and 3,400 core environments and reaches 35.2% on Terminal-Bench 2.1, whereas the middle and high cost tiers of SCOPE accept 346 and 691 parents with 1,138 and 2,293 core environments and reach 30.6% and 33.8%. The source trend therefore describes the nested sweep, and the cost comparison describes separately constructed collections under common ceilings, so each is read within its own selection protocol. Where Unseen is reported, the sampler study uses the full-family balanced mean over all 980 retained members, whereas the other sweeps use the sentinel-pair mean over 600 states. Both weight parents equally but use different member sets, so the 36.4% that both report for the shared family-aware checkpoint belongs to two distinct estimands.

- **Independent sources** uses domain-stratified, hash-nested sets of 128, 256, 512, and 1,024 core parents with all their retained members, shared auxiliary data, and the same starts, with 421, 842, 1,694, and 3,400 core environments and a mean realized k of 2.3.
- **RL compute** trains on the fixed collection for 16M, 32M, and 64M learner-generated tokens.
- **Construction cost** follows the tiers of Table 20. Methods may accept different members and counts, and Trace-Feedback is run at the middle tier only.
- **Family depth** forms nested member subsets under caps of 1, 2, and 4 on the same 1,024 parents by the original priority order, without resampling to fill a cap, which gives 2,048, 2,909, and 3,400 core environments with a mean realized k of 1.0, 1.8, and 2.3.
- **Point cap** compares, under the same source pool, accounting, and middle-tier budget, semantic caps of one, two, and all identified points per source, at most six, together with one uniformly chosen legal step per source. The ranked point list is frozen before construction, multi-point slots are allocated by a preregistered round-robin over the same proposal budget and variant cap, and each point group keeps its own real history and its own witness. Identical initial states are counted once, and training normalizes within-parent weight so that additional points do not multiply source weight.
- **Sampler** trains on the same 1,024-parent environments, rewards, and budget and changes only the loader. It compares family-aware sampling with three flat shuffles, one that balances the original and variant mass only in aggregate, so that the variant-side weight of a parent grows with its family size, one that samples unique environments uniformly, and one that restores equal parent weights within the same flat loader.

Table 22: Scaling axes with Qwen3.5-9B, as Terminal-Bench 2.1 success (%) at the 32M budget unless stated. The source counts give 421/842/1,694/3,400 core environments, and the SCOPE cost tiers accept 173/346/691 parents with 571/1,138/2,293 core environments (Table 20).

Axis	Conditions	SCOPE (Ours)	Random	Feedback
Sources	128/256/512/1,024 parents	24.6/28.2/31.8/35.2	19.4/21.6/24.3/26.4	21.3/23.7/26.1/28.4
RL compute	16M/32M/64M tokens	30.4/35.2/39.4	23.1/26.4/29.5	25.0/28.4/31.7
Cost	low/middle/high tier	27.5/30.6/33.8	22.1/24.6/26.2	23.0/25.3/27.8
Family depth	cap 1/2/4	32.3/34.1/35.2	–	–
Point cap	semantic 1/2/all; random 1	30.6/29.4/28.5; 26.7	–	–
Sampler	family/flat bal./unbal./par.	35.2/32.8/30.6/35.0	–	–

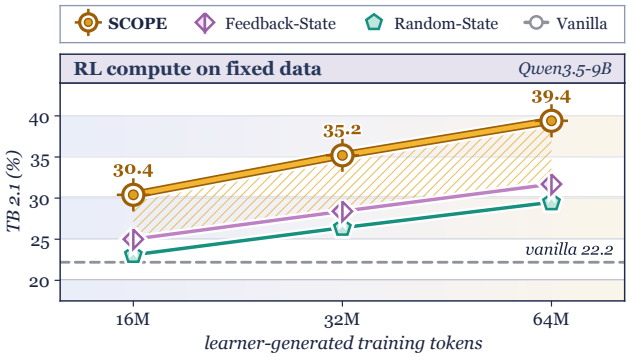


Figure 7: Terminal-Bench 2.1 success of Qwen3.5-9B against RL compute on the fixed collection.

G.3 Framework Analysis ($\mathcal{RQ}4$)

Construction Ablation. Figure 8 and Table 23 give the six configurations behind the first block of Table 2, trained at the 32M budget with one variant per parent. From 1,200 screened parents, 1,083 to 1,158 are valid for each configuration and 1,038 for all six, and the calipers remove 85 of these (8.2%), which leaves 953 matched parents. With two environments per parent plus the 400 background and 200 control environments, each configuration trains on 2,506 unique environments, so this study and its support are reported separately from the main 4,000-environment collection. Each configuration selects members in its own fixed order under a shared allowance of 32 candidates per parent, half blind and half trace-visible, and a source point frozen before construction. Dropping a condition means not requiring it rather than forcing it to fail, so each ablation relaxes one requirement. Without H, differences may be exposed in the prefix, without F, the frozen continuation may still succeed, and without I, no common repair is verified. The dropped-H configuration keeps each member's real history rather than copying observations. Matching constrains reference-call and active-time ratios to $[0.5, 2]$ separately at roots and frontiers and the semantic edit-factor gap to at most one, without using learner success or reselecting candidates, and 2,076 member observations per configuration become 1,906 after matching. Table 24 reports the diagnostics, where the largest standardized difference after matching is 0.1, a diagnostic line rather than proof of matched latent difficulty.

Table 23: Construction ablation with Qwen3.5-9B at the 32M budget on 953 matched parents (%). A dropped condition is not required rather than forced false.

Configuration	H	F	I	TB 2.1	Unseen	Long
Random	✗	✗	✗	20.4	27.2	18.2
Feedback	✗	✗	✗	22.5	29.3	21.1
w/o H	✗	✓	✓	23.6	30.4	22.3
w/o F	✓	✗	✓	22.2	29.2	21.1
w/o I	✓	✓	✗	21.3	28.1	20.2
SCOPE (Ours)	✓	✓	✓	**29.1**	**35.4**	**28.4**

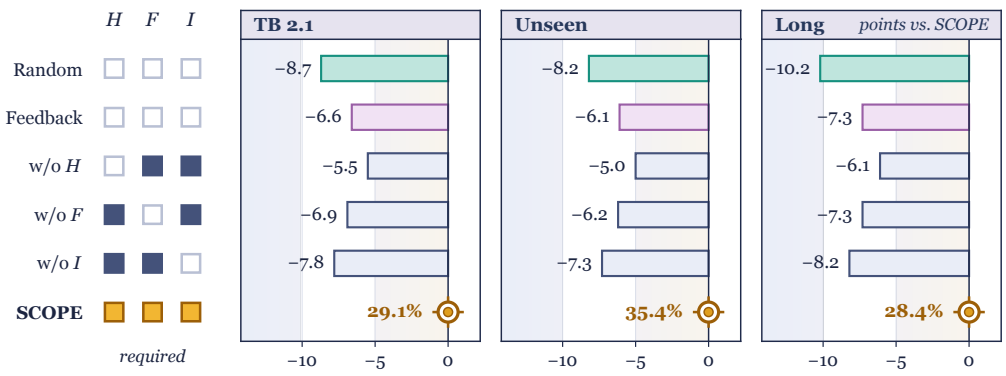


Figure 8: Construction ablation on 953 matched parents, as the conditions each configuration requires (left) and success relative to SCOPE in points (right).

Table 24: Matching diagnostics of the construction ablation. Overlap is the share of the 953 matched parents on which a configuration selects the same candidate as SCOPE, and SMD is the largest absolute standardized mean difference against SCOPE over roots and frontiers, computed with pooled sample variances before rounding.

Configuration	Parents	Shared	Overlap (%)	SMD, calls	SMD, time
Random	953	119	12.5	0.1	0.1
Feedback	953	216	22.7	0.0	0.1
w/o H	953	510	53.5	0.1	0.0
w/o F	953	480	50.4	0.0	0.1
w/o I	953	300	31.5	0.1	0.1
SCOPE (Ours)	953	953	100.0	–	–

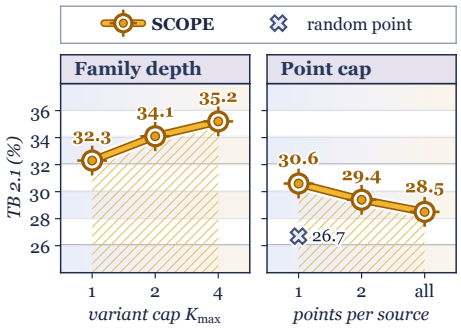


Figure 9: Sensitivity analysis of SCOPE.

Sensitivity Analysis. We analyze the sensitivity of SCOPE to its two caps, the variant cap $K_{\max}$ and the number of expansion points per source (Figure 9). For $K_{\max}$, success rises with a diminishing step as the cap grows from 1 to 4 on the same parents (32.3% → 34.1% → 35.2%). For the point cap, success falls as it grows from one to two to all identified points under one construction budget (30.6% → 29.4% → 28.5%), and a uniformly chosen single point reaches 26.7%. Under the fixed budget, semantic single-point selection retains 346 origins, against 268 for two points and 237 for all identified points. The decline is consistent with a trade-off between breadth and depth in the tested configurations, although it does not identify the number of sources as the only cause. Considering both, we expand each source at one point with up to four variants.

Sampler Weights. The flat loader that restores equal parent weights reaches 35.0% on Terminal-Bench 2.1, 2.2 points above aggregate balancing (95% interval [0.7, 3.7]) and 0.2 points below

family-aware sampling ([−1.1, 0.6]), within the preregistered equivalence margin of 1.5 points. On the full-family balanced Unseen mean, the corresponding differences are 2.3 [1.0, 3.5] and −0.1 [−0.8, 0.5]. At the 32M target, family-aware sampling visits 63, 61, and 65 of the 1,024 core parents across the three seeds, while the flat parent-equal control visits 127, 121, and 131, so each loader is compared at the same token target together with the data exposure its batching induces. Under this budget and implementation, restoring parent weights thus recovers most of the gap, and batching families together adds little once parent weights match.

Information Interventions. Table 25 gives all four arms behind Figure 5. The 100 Mechanism parents are preregistered, isolated from the motivation roster, and not selected by any model's outcome. Each contributes its original and sentinel state, and each of the five model and method conditions is evaluated on both members under all four conditions with three seeds and five repeats, which gives $100 \times 2 \times 5 \times 4 \times 3 \times 5 = 60{,}000$ executions. Each learner continues from a common reference prefix under natural continuation or one of three post-check conditions, which use equivalent physical states and show the relevant result, neither result, or only the irrelevant result under the same display rules. For a comparator b, the interaction contrast is

$$D_b = \left(p_{\texttt{SCOPE},\mathrm{nat}} - p_{b,\mathrm{nat}}\right) - \left(p_{\texttt{SCOPE},\mathrm{rev}} - p_{b,\mathrm{rev}}\right), \tag{9}$$

which equals 15.2 points against Random-State and 7.9 against Trace-Feedback with Qwen3.5-9B, and 9.4 against Trace-Feedback with Qwen3.5-4B. Within each method, revealing the result helps SCOPE by 8.5 points over natural continuation with Qwen3.5-9B, against 16.4 for Trace-Feedback and 23.7 for Random-State. D_b is a diagnostic of interaction rather than an identified mediation fraction, since reveal against mask fixes the physical post-check state and isolates the presence of the result in context, whereas reveal against natural also includes the physical effect of running the checks. Realized post-check states are tied to each model and repeat's actual preparation rather than to one shared snapshot.

Table 25: Success rate (%) under information interventions on the Mechanism suite, continuing from a common reference prefix. Best in **bold**, runner-up underlined.

Method	Natural	Reveal	Mask	Irrelevant
Qwen3.5-4B				
Trace-Feedback	38.4	55.8	39.1	38.7
SCOPE **(Ours)**	**49.9**	**57.9**	**49.0**	**49.6**
Qwen3.5-9B				
Random-State	43.1	66.8	44.0	43.4
Trace-Feedback	52.7	69.1	52.4	53.1
SCOPE **(Ours)**	**62.5**	**71.0**	**61.3**	**62.2**

Selectivity. Each of three state controls holds two complete root states per Mechanism parent (Table 26). The relevant control is the original and sentinel pair of the main Mechanism suite, evaluated with identical checkpoints, random streams, budgets, and scoring, so its executions and its 6.1-point difference with Qwen3.5-9B against Trace-Feedback are those of Table 1. The irrelevant control keeps that original state and changes only independently verified task-irrelevant facts in the second state. The fact-given control keeps the two physical states and makes the relevant fact public, which requires new executions with the changed instruction. A separate roster of 400 single-environment tasks that require no discovery runs the same methods. These two-member controls are distinct from the root and frontier training mixture. The three controls comprise 45,000 logical trials, of which 22,500 reuse the main executions, namely the relevant pair and the original branch of the irrelevant control, and 22,500 are new. The token ratio compares total generated tokens, including failed and timed-out runs. Against Trace-Feedback, the advantage of SCOPE falls from 6.1 points on relevant changes to 0.2 on irrelevant changes and 1.1 when the fact is given, with token ratios near 100%. On the no-discovery tasks, the difference is 0.3 points with an interval of [−1.1, 1.7] and a token ratio of 100.6% with an upper end of 102.4%, both inside the preregistered margins. Against Random-State, gaps of 4.0 points on irrelevant changes and 4.4 points on the no-discovery tasks remain, in line with the general gap between the two methods in Table 1, so the selectivity conclusion rests on the comparison with Trace-Feedback.

Table 26: Selectivity of SCOPE. Success differences are in points and token ratios in percent of the comparator, both with 95% source-cluster intervals. The no-discovery rows are judged against a margin of −3.0 points and a ratio of 110.0%.

Condition	Success difference	Token ratio (%)	Parents
Qwen3.5-9B vs. Trace-Feedback			
Relevant state change	+6.1 [3.0, 9.3]	96.9 [93.6, 100.4]	100
Irrelevant state change	+0.2 [−1.8, 2.2]	101.1 [98.8, 103.5]	100
Relevant fact given	+1.1 [−0.9, 3.1]	100.8 [98.5, 103.2]	100
No discovery required	+0.3 [−1.1, 1.7]	100.6 [98.9, 102.4]	400
Qwen3.5-9B vs. Random-State			
Relevant state change	+11.2 [7.8, 14.7]	93.4 [90.1, 96.9]	100
Irrelevant state change	+4.0 [1.7, 6.4]	96.0 [93.2, 98.8]	100
Relevant fact given	+3.9 [1.2, 6.7]	97.0 [94.0, 100.0]	100
No discovery required	+4.4 [2.7, 6.1]	97.1 [95.1, 99.3]	400
Qwen3.5-4B vs. Trace-Feedback			
Relevant state change	+5.6 [2.7, 8.6]	98.2 [94.3, 102.0]	100
Irrelevant state change	+0.3 [−1.9, 2.6]	100.6 [98.0, 103.4]	100
Relevant fact given	+1.2 [−1.1, 3.4]	99.7 [97.0, 102.7]	100
No discovery required	+0.4 [−1.3, 2.0]	100.5 [98.5, 102.7]	400

Behavioral Metrics. Table 27 collects the behavioral outcomes behind §3.4. The two rosters are disjoint, keep separate denominators, and hold two state members per parent, so with three seeds and five repeats the pre-commitment study plans 3,000 executions per method and the shared-error study 1,800. A common start is shared across methods for the same parent and member. On the 100 pre-commitment families, the first erroneous commitment rate falls from 32.2% to 19.9% at a monitoring coverage of 95.4% and 95.1%, and final success rises from 48.2% to 62.4%. On the 60 families that start from a shared recoverable error state, success without propagation rises from 35.0% to 55.2%, and among monitored runs the error is repaired in 80.5% of cases against 62.3%.

Table 27: Behavioral metrics of Qwen3.5-9B (%). The first erroneous commitment rate is conditional on monitoring coverage, and the joint outcomes of the shared error state sum to the planned denominator with unknowns retained.

Metric	Random-State	SCOPE (Ours)
Pre-commitment families (100)		
First erroneous commitment ↓	32.2	**19.9**
Final success ↑	48.2	**62.4**
Monitoring coverage	95.4	95.1
Shared recoverable error state (60)		
Success without propagation ↑	35.0	**55.2**
Success with propagation	10.2	7.9
Failure without propagation	25.1	17.2
Failure with propagation ↓	24.9	**14.9**
Unknown	4.8	4.8
Repair of the error when monitored ↑	62.3	**80.5**
Success, lower and upper bound	45.2–49.9	63.1–67.9

Decision Position. Table 28 reports the scores behind Figure 3 (right). The 60 source blocks each hold three genuinely different complete tasks with their own references rather than three renamed forks of one trajectory, and each task has two state members, each evaluated from its root and from its common reference checkpoint. At each position, a method and start therefore cover 60 parents, 2 members, 3 training seeds, and 5 repeats, or 1,800 executions, and the study holds 21,600 executions in total. Each task has seven substantive stages with a checkpoint after stage 1, 3, or 5, and the relevant fact first affects processing in the following stage, so that early checkpoints have real history and late checkpoints retain substantive work. The root start measures complete-task

success. The common reference checkpoint presents the same state and remaining budget to both methods and measures the conjunction of the prespecified target requirements, the primary outcome, while retaining final success, so that it does not condition on each agent's own arrival at the later stage. The two starts have separate denominators and are not averaged, and pairing is by source block rather than by task.

Table 28: Success rate (%) of Qwen3.5-9B on 60 matched blocks of seven-stage workflows by the position of the first unresolved decision.

Position	Primary outcome			Final success	
	Random-State	SCOPE	Change	Random-State	SCOPE
Root, complete-task success					
Early (after stage 1)	27.3	**41.5**	↑14.2	–	–
Middle (after stage 3)	23.4	**37.2**	↑13.8	–	–
Late (after stage 5)	19.1	**33.5**	↑14.4	–	–
Common checkpoint, target-requirement success and final success					
Early (after stage 1)	45.3	**57.5**	↑12.2	40.7	**52.8**
Middle (after stage 3)	42.2	**55.4**	↑13.2	37.9	**51.2**
Late (after stage 5)	40.3	**54.5**	↑14.2	36.1	**50.2**

Consequential Errors. Figure 10 shows the joint outcomes on the 60 common-error families, which start from a prespecified, actually induced, and reference-recoverable error state shared across methods, disjoint from the 100 pre-commitment families, with one preregistered intervention position per reference. Commitment, consumption, and propagation are defined by real object versions and public semantics rather than by command keywords, a temporary unfinished file is not an erroneous commitment, and observation-only monitors neither prompt nor intercept the agent. Legal paths outside monitoring coverage keep normal final verification, with coverage reported separately. The five categories sum to the planned denominator, with the 4.8% unknowns retained rather than counted as failures. On the pre-commitment families, the first erroneous commitment rate is conditional on monitoring coverage, whereas final success uses the complete planned denominator.

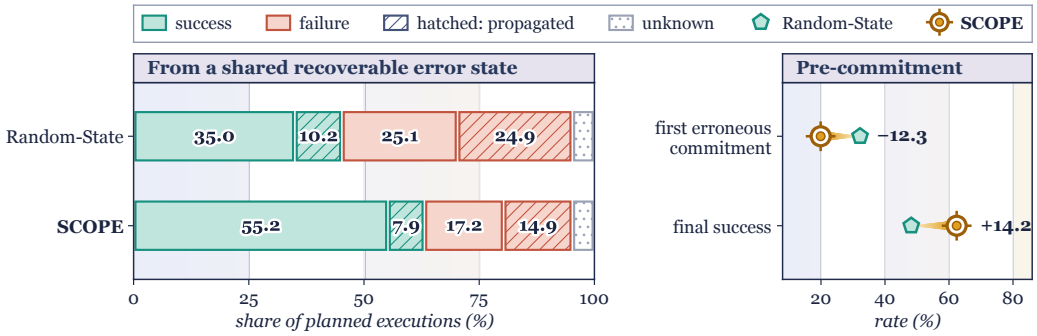


Figure 10: Behavior of Qwen3.5-9B from a shared recoverable error state (left, over planned executions) and on pre-commitment families (right, erroneous commitment over monitored runs and final success over planned executions).

Auxiliary SFT. Table 29 reports the auxiliary study of Appendix E.4. The direct and inspect SCOPE corpora use the same parents and states, with successful trajectories selected by their observed demonstration form. *Direct* demonstrations continue without first revealing the relevant fact, whereas *inspect* demonstrations obtain that fact through ordinary tools before adapting. The corpora share their task support and training budget, while their generated solution content and lengths can differ. At every scale, inspect demonstrations outperform direct ones on both suites, for example 30.8% against 28.8% on Terminal-Bench 2.1 with Qwen3.5-9B, and both outperform SFT-Trace. Continuing SFT-SCOPE-inspect with RL improves both suites further, while direct RL gives the

highest scores among the tested routes at all three scales. This ordering is specific to the shared data and the separately accounted SFT and RL budgets.

Table 29: Auxiliary SFT study at three scales (%). The scales share the frozen teacher corpora and parent support, and Vanilla and direct RL reuse the main-evaluation checkpoints. Best in **bold**, runner-up underlined.

Route	Task-specific SFT	RL budget	TB 2.1	Unseen
	Qwen3.5-4B			
Vanilla	none	none	15.7	17.3
SFT-Trace	Trace corpus	none	18.9	23.7
SFT-SCOPE-direct	direct corpus	none	20.1	25.0
SFT-SCOPE-inspect	inspect corpus	none	22.2	27.3
Direct RL-SCOPE	none	64M	**28.2**	**33.0**
SFT-SCOPE-inspect→RL	inspect corpus	64M	26.8	31.1
	Qwen3.5-9B			
Vanilla	none	none	22.2	23.1
SFT-Trace	Trace corpus	none	27.3	30.2
SFT-SCOPE-direct	direct corpus	none	28.8	31.0
SFT-SCOPE-inspect	inspect corpus	none	30.8	34.1
Direct RL-SCOPE	none	64M	**39.4**	**42.1**
SFT-SCOPE-inspect→RL	inspect corpus	64M	37.4	40.1
	Qwen3.5-27B			
Vanilla	none	none	43.1	41.7
SFT-Trace	Trace corpus	none	46.4	47.0
SFT-SCOPE-direct	direct corpus	none	47.6	48.8
SFT-SCOPE-inspect	inspect corpus	none	49.5	51.7
Direct RL-SCOPE	none	64M	**54.2**	**57.1**
SFT-SCOPE-inspect→RL	inspect corpus	64M	53.0	55.8

H LIMITATIONS

SCOPE applies to tasks that can be rebuilt faithfully, compared under a fixed visible protocol, and verified independently, and certifying a family costs more than generating an unconstrained variant, although the gain holds at matched construction cost (Appendix G.2). Expansion points are chosen by LLM review and outcomes are judged by verifiers, and both can err. Every retained family is therefore certified by execution as well, and in the audit of Table 7, 98.0% of the points pass an independent review while the verifiers misjudge 1.0% of near-misses and 0.9% of legal alternatives. Construction relies on an LLM teacher, and a frontier presents the teacher's reference prefix as context, so we do not describe the main route as learning without teacher help, although every constructed baseline shares the same teacher, references, and two starts. The internal suites are built by our own pipeline, and Unseen pairs each original with a variant certified by the same gates, so they may favor SCOPE. Our external claim therefore rests on Terminal-Bench 2.1, where the gains hold at all three scales. Finally, all learners come from one model family, most analyses beyond the main comparison use Qwen3.5-9B, and the motivation study measures the adaptation gap within its preregistered roster rather than its prevalence across terminal tasks.